CERT

CERT RESEARCH ANNUAL REPORT 2009

Software Engineering Institute | Carnegie Mellon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2009**	2. REPORT TYPE	3. DATES COVERED **00-00-2009 to 00-00-2009**
4. TITLE AND SUBTITLE **CERT Research Annual Report 2009**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**82**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The primary goals of the CERT® Program are to ensure that appropriate technology and systems management practices are used to resist attacks on networked systems and to limit damage and ensure continuity of critical services in spite of attacks, accidents, or failures.

CERT is part of the Software Engineering Institute (SEI), a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University. The SEI advances software engineering and related disciplines to ensure systems with predictable and improved quality, cost, and schedule.

This report describes how CERT research advanced the field of information and systems security during the 2009 fiscal year.

To download a PDF version of this annual report, go to http://www.cert.org/research/2009research-report.pdf

TABLE OF CONTENTS

CERT Research Vision

Today we live in a world in which the threat of cyber attacks is ever-growing, and where threats from unknown sources are dynamic and constantly changing. It is seldom that a week goes by when articles on cyber security are not prominent in technical publications and popular media. It is the CERT Program's mission to identify, develop, and mature, and broadly transition new technologies, system development practices, and system management practices that enable informed trust and confidence in using information and communication technology.

Our stakeholders include the U.S. Department of Defense, the Department of Homeland Security, other U.S. federal agencies, state and local governments, and other operators of infrastructures critical to the national defense, cyber security, and the national economy; the providers of information communications technologies (ICT) and services that support these system and network operators; the software development community; and computer security incident response teams with national responsibilities.

The overall goal of our program is improved practices and technologies that are widely understood and routinely used to protect, detect, and respond to attacks, accidents, and failures on networked systems. Better informed, trained, and equipped people will produce better systems that will be better managed to reduce operational risk and the impact of cyber attacks.

Our research strategy has been to build and maintain a technical center of excellence that uses its operational experience and expertise to look across the entire software life cycle (from requirements through development, deployment, operations, and maintenance) to

- identify new technologies, development practices, and management practices that would significantly improve networked systems security and enterprise resiliency
- mature these technologies and practices
- apply these technologies to meet the needs of the program's stakeholders
- transition these technologies into widespread use

The transition activity pays special attention to education and training and recognizes the critical need to develop an international workforce skilled in software and information assurance.

Our objectives include the following

Objective one: Identify and establish techniques and approaches needed to embed software and system security assurance in all aspects of the system development life cycle.
The program is identifying and establishing techniques and approaches needed to embed software and system assurance in all aspects of the system development life cycle. We are transitioning these methods into widespread use with traditional methods and recently developed approaches, such as DHS's Build Security In web site, the Resiliency Engineering Framework, and the SEI's Virtual Training Environment.

Objective two: Improve the effectiveness of the international intrusion analysis and response team community.
Improve the effectiveness of the intrusion analysis and response team community by developing a commonly used infrastructure of policies, practices, and technologies that reduces the impact of cyber attacks by supporting rapid identification, collaborative analysis, and forensic investigation. Integrate insider threat research analysis and findings into new technologies for the detection and prevention of malicious insider activity. Create and transition incident response tools that provide commercially unavailable capabilities. Support the development of teams with national responsibility.

Objective three: Develop a workforce skilled in secure cyber operations.
Develop a workforce skilled in secure cyber operations by developing curricula on network operations, information assurance, incident response, and forensics analysis for executives, managers, educators, software engineers, and network administrators; and developing curricula for cyber forensics for law enforcement investigators as well as front-line system operators. Transition the curricula using traditional methods to academic institutions and other stakeholders as well as through innovative approaches such as the CERT Virtual Training Environment (VTE) and XNET.

Objective four: Improve the survivability and resiliency of the program's stakeholders.
Improve the survivability and resiliency of the program's stakeholders by developing and transitioning disciplined improvement approaches including improvement models (such as the Resiliency Management Model), security assessment methods, compliance validations, and threat analysis, modeling, and mitigation techniques. Provide practical guidance for identifying and managing insider threats and other risks.

The sophistication of technology and attack methods continues to evolve rapidly, and we must develop mitigation strategies and solutions at the same pace. The need to maintain constant vigilance and awareness of the threat landscape is important, and organizations need to keep security awareness at the forefront. Security policies must be dynamic and adapted as necessary to address the evolution of the threats. Striving to achieve these objectives requires our own best efforts, as well as cooperation and collaboration within the community we serve.

The following pages highlight a sample of our work, and we will continue to promote and rely on cooperation and collaboration within the security community. We welcome participation by organizations that share our belief that the networked environment of the future can provide safe and secure computing for all participants in the global information community.

Richard Pethia
Director, CERT
Software Engineering Institute
Carnegie Mellon University

Executive Summary

The work of the CERT Program at Carnegie Mellon University's Software Engineering Institute includes technologies and methods for

- eliminating security flaws and vulnerabilities in systems
- preventing intrusions from occurring
- identifying intrusions that have occurred
- preserving essential services when systems have been penetrated and compromised
- providing decision makers with information required for network defense

We recognize the importance of multiple strategies for prevention and detection of and recovery from cyber security attacks, and the CERT Program has been designed to address a broad spectrum of security technology research, development, and transfer.

In our research activities, the goal is to replace informal methods with precise software and security engineering. In our technology development work, we create software and security standards, technologies, and automation. In technology transfer, we work with clients to incorporate results into key acquisition and development projects. We also provide training and materials, such as books and articles, to support technology transfer.

While all these elements are necessary to achieve success, the focus of this report is on CERT's research work. Our research agenda is driven by the need to develop theoretical foundations and engineering methods to help ensure the security of critical systems and networks. We believe the projects described in this report are essential elements of this agenda. Abstracts are provided here for our major research projects, followed in the report by more detailed descriptions of the projects. Additional research activities, publications, and technical leadership activities are also described.

2009 Research Report Abstracts

Applying Function Extraction (FX) Techniques to Reverse Engineer Virtual Machines

Malware that has been protected by intruders using virtual machine technology, or virtualized malware, represents a significant challenge to fast and effective reverse engineering. A virtual machine is a program that emulates one computer's architecture on another architecture. Intruders can create a "one-off" virtual machine and translate their malware to its unique architecture. However, virtualized malware must eventually be translated for execution on target machines such as the Intel IA-32 architecture, for which the meaning, or semantics, of each instruction is well known. To analyze virtualized malware, researchers applied CERT's Function Extraction (FX) techniques for software behavior computation to understand the functional effect of sequences of virtual machine instructions. This approach permitted undoing of most of the virtual machine encoding, resulting in a standard IA-32 program suitable for reverse engineering with existing tools and techniques.

A Probabilistic Population Study of the Conficker-C Botnet

Botnet size is often reported as a number of IP addresses, but the link between IP addresses and infected machines is more complicated than a simple one-to-one relationship. To count the number of infected machines when we have only an aggregated view of a botnet, we suggest building a precise probability model of the observable behavior of a single machine, and applying that model to the aggregates to obtain a population estimate. As an example, we build a probability model of the peer-to-peer (P2P) scanning activity of the Conficker-C botnet, and we use this model to estimate the number of active infected machines per hour over a two-month window.

Catching IPv6 Tunneled in IPv4

Internet Protocol version 6 (IPv6) is the long term solution for the depletion of IPv4 addresses and has been increasing in deployment over the past decade. To ease the adoption of IPv6, automatic tunneling protocols (i.e. 6to4, ISATAP, Teredo) will encapsulate IPv6 information inside IPv4 with minimal user effort. These protocols allow for IPv6 to run in networks regardless of the preparedness or awareness of the network administrators. In this work, we use features of these tunneling protocols to identify IPv6 traffic that is masquerading as IPv4.

Finding Malicious Activity in Bulk DNS Data

The Domain Name System is a vital component of the Internet, and nearly every transaction on the Internet uses it. It contains a wealth of Network Situational Awareness information that can be used to discover malicious traffic. This report describes specific techniques to detect certain types of malicious traffic. These techniques have been developed through analyzing a large amount of DNS traffic data. CERT has developed specific tools that apply these techniques in an ongoing way. Future research will include enhancing the developed tools, developing new techniques and tools to work with known malicious patterns, and discovering new malicious patterns.

Function Extraction for Malicious Code Analysis

As the quantity and sophistication of malicious code continues to grow, automation support for analysis becomes more important to keep pace with the scope and scale of the problem. To help address this need, CERT originated the technology of Function Extraction (FX) for automated computation of software behavior (including malware) with mathematical precision to the maximum extent possible. Intruders often obfuscate malware packages to make analysis more difficult by inserting massive amounts of arbitrary jumps in the code that thwart control flow tracing, and by inserting blocks of no-op code that have no functional effect but must nevertheless be analyzed. A specialization of FX technology in the Function Extraction for Malicious Code (FX/MC) system is designed to address these obfuscation problems. FX/MC eliminates control flow obfuscation caused by arbitrary jumps by transforming the code into function-equivalent structured form, and applies behavior computation to detect and eliminate blocks of no-op code. These operations can result in smaller and less complex malware for analysis.

Function Hashing for Malicious Code Analysis

Comparing samples of malicious code to each other continues to be an interesting field of research because it allows reverse engineers to save time otherwise spent analyzing malicious code they have looked at before. In our function hashing research, we have laid a foundation for code comparison that dramatically reduces the size of the code similarity problem facing malicious code analysts. Our approach was to use high-confidence cryptographic hashing algorithms to identify duplicated code in approximately four million files in the CERT Artifact Catalog. Our results indicate that over 1.5 billion function instances can be meaningfully represented by less than 40 million unique hashes. This reduction in the scale of the challenge can lead to improved correlation of similar malware based on their functionality.

Metrics for Evaluating Network Sensor Placement

Network sensors collect, detect, or help mitigate malicious activity on network traffic data. Metrics for evaluating their benefits at a given location can be improved. The improvements in the metrics help in prioritizing sensor deployment and improve network security for an organization, given a budget constraint. An existing metric baseline was improved by two implementation approaches. One approach consisted of examining the metric along with its weights, and then suggesting improvements. Based on expert opinion, suggestions were made to modify the existing metric to improve its accuracy. The other approach consisted of identifying new measures that could be included to form a more comprehensive metric. Four new measures were constructed to augment the metric using network data. The result is a new metric that should better reflect the benefits from deploying a sensor at any particular location.

Modeling Insider Theft of Intellectual Property

A study conducted by the CERT Program at Carnegie Mellon University's Software Engineering Institute analyzed hundreds of insider cyber crimes across U.S. critical infrastructure sectors. Follow-up work involved detailed group modeling and analysis of 48 cases of insider theft of intellectual property. In the context of this paper, insider theft of intellectual property for business advantage includes incidents in which the insider's primary goal is stealing confidential or proprietary information from the organization with the intent to use it to take to a new job, to get a new job, or to start a business. It does not include cases in which insiders sell an organization's information. This paper describes general observations about, and a preliminary system dynamics model of, this class of insider crime based on our empirical data. This work generates empirically-based hypotheses for validation and a basis for identifying mititgative measures in future work.

Rayon: A Unified Framework for Data Visualization

Data visualization summarizes large volumes of data and represents this data pictorially. Data visualization is used in a wide variety of applications, but visualization techniques that are effective in one application can often be used as well or better in another application. When organizations depend on good data visualization, a unified visualization capability will often increase that effectiveness; this is especially important if an organization relies on internal experts to create new visualization techniques appropriate to their environment. The Rayon visualization toolkit was developed to augment large scale network analytic information, and to improve the visualization capability and productivity of analytic operations by making it possible to share visualization techniques between applications.

Source Code Analysis Laboratory

CERT established the Source Code Analysis Laboratory (SCALe) to address the problem of insecure code, that is, code that contains known vulnerabilities. The purpose of SCALe is to investigate approaches to developing systems that are certifiably free from known vulnerabilities.

SQUARE: Security Quality Requirements Engineering

Through the SQUARE Project, CERT researchers have developed an end-to-end process for security requirements engineering to help organizations build security into the early stages of the production life cycle. The SQUARE methodology consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. The process has been baselined and transitioned into practice. A robust tool was released this year and one to two day training courses are available, as well as educational materials for download. SQUARE for acquisition (A-SQUARE) and for privacy (P-SQUARE) are under active development.

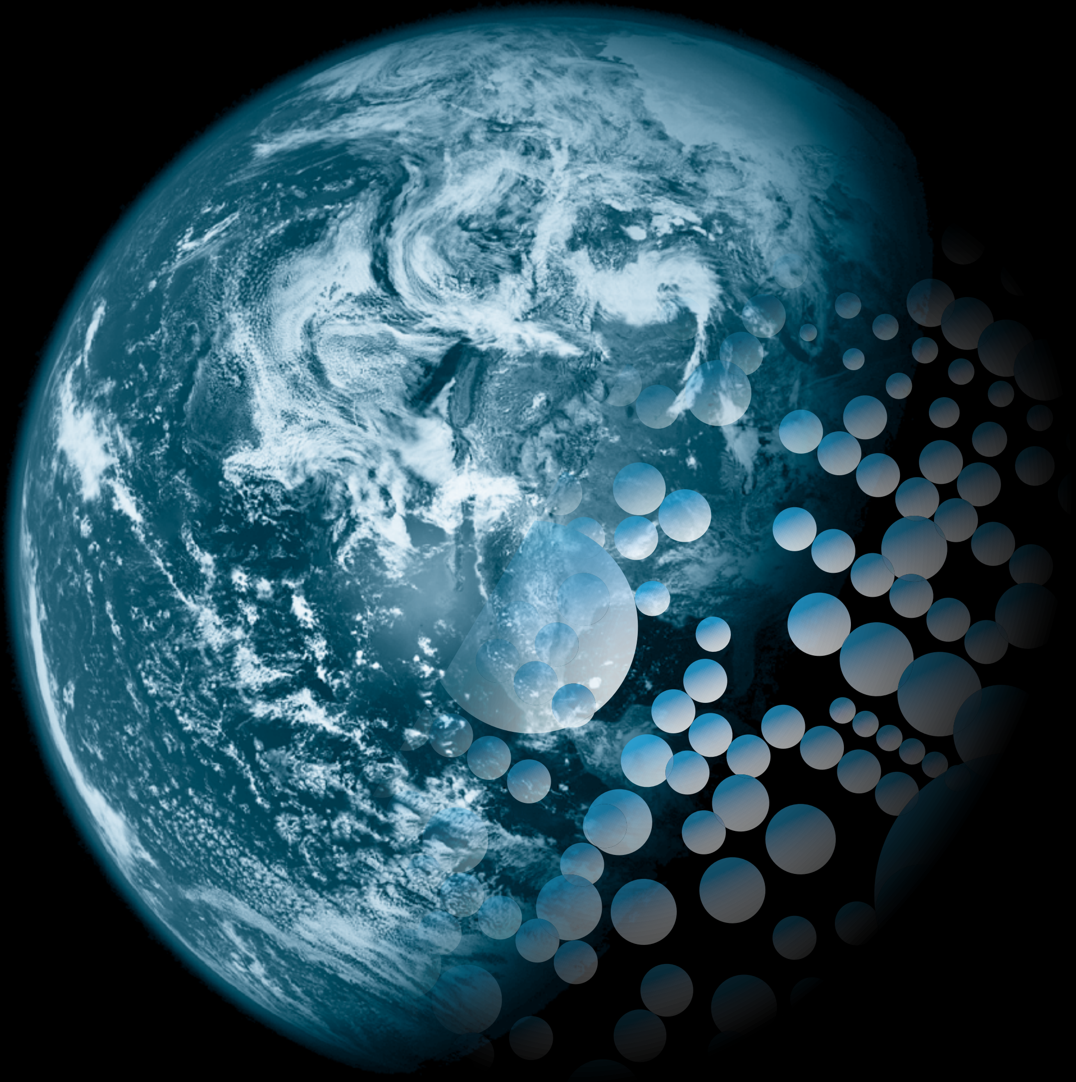
Mark Pleszkoch

Stacy Prowell

Cory F. Cohen

Jeffrey S. Havrilla

Applying Function Extraction (FX) Techniques to Reverse Engineer Virtual Machines



Applying Function Extraction (FX) Techniques to Reverse Engineer Virtual Machines

Introduction

Malware that has been protected by intruders using virtual machine technology, or virtualized malware, represents a significant challenge to fast and effective reverse engineering. A virtual machine is a program that emulates one computer's architecture on another architecture. Intruders can create a "one-off" virtual machine and translate their malware to its unique architecture. However, virtualized malware must eventually be translated for execution on target machines such as the Intel IA-32 architecture, for which the meaning, or semantics, of each instruction is well known. To analyze virtualized malware, researchers applied CERT's Function Extraction (FX) techniques for software behavior computation to understand the functional effect of sequences of virtual machine instructions. This approach permitted undoing of most of the virtual machine encoding, resulting in a standard IA-32 program suitable for reverse engineering with existing tools and techniques.

Problem Addressed

Reverse engineering relies on tools and techniques developed for common computer architectures, such as the 32-bit Intel architecture (IA-32). Programs compiled for less-common architectures will consequently have fewer readily available tools for the reverse engineer to apply. The worst case is a single-use, randomly-generated architecture that has never been seen before. No standard tools and techniques will be readily available, and the first step of reverse engineering will be to understand the architecture.

A *virtual machine* is a computer program that emulates a machine. For example, the Java Virtual Machine (JVM) emulates a stack-based architecture and executes programs compiled for this architecture. A virtual machine allows emulation of one computer's architecture on another, different architecture, as illustrated in Figure 1.

Virtual machine protectors exploit differences in architecture to prevent reverse engineering compiled programs. By creating a "one off" virtual machine and then translating existing programs to the virtual machine's architecture, the virtual machine protector makes the reverse engineer's job significantly harder.

Malware that has been protected using virtual machine technology, or *virtualized* malware, represents a significant challenge to fast and effective reverse engineering.

Research Approach

Despite being written for a specialized virtual architecture, programs must still operate on a well-known platform such as IA-32. That is, there must be a point at which each virtual instruction is translated into the IA-32 instruction(s) that accomplish the desired effect. This translation typically happens while the program is running, on a per-instruction basis.

This is the fundamental weakness of the virtual machine strategy: malware authors cannot change the physical architecture where the program will ultimately execute. The meaning, or *semantics*, of each instruction on the physical architecture is fixed and well known. Despite this, it is not enough to simply capture the instruction stream produced as the program is executed, for several reasons:

- Execution results in a stream of hundreds of thousands of instructions, and quickly overwhelms the analyst.
- The instruction stream captured during execution may not be representative of all program functionality.
- The instruction stream may be intentionally *obfuscated* to make it harder to understand.

Despite the virtual machine encoding, the end program must accomplish the same *functional* effect, or it will not be able to accomplish its mission.

Our approach was to apply CERT's Function Extraction (FX) techniques for software behavior computation [1] to reason about the functional effect of sequences of virtual machine instructions. By examining the overall functional effect of sequences of virtual instructions we were able to detect when a sequence of virtual instructions accomplished the same effect as some IA-32 instruction, and then to record that *decoded* IA-32 instruction. In this manner we were able to undo most of the virtual machine encoding that had been performed, obtaining a standard IA-32 program suitable for reverse engineering with existing tools and techniques.

In more detail, a specific sample of virtualized malware was presented to the team for analysis. Through traditional reverse engineering techniques the dispatch loop of the virtual machine engine was discovered, allowing determination of the instruction decode mechanism. Each virtual instruction corresponded to a sequence of IA-32 instructions. Since the current FX technology can reason about sequences of IA-32 instructions, we were able to automatically obtain the semantics of each virtual instruction, as depicted in Figure 2.

Next we were able to apply the semantics of the virtual instructions to obtain the overall semantics of sequences of virtual instructions, and then, again using the semantics of the IA-32 instruction set, to recognize when sequences of virtual instructions performed the same effect as a sequence of IA-32 instructions. This allowed most of the virtualized program to be "decoded" to reveal the original program, prior to virtualization, as illustrated in Figure 3.

Benefits

By applying FX technology to the problem of virtual machine protectors we addressed a serious threat to software reverse engineering and demonstrated the effectiveness of the FX technology developed by CERT.

Virtualization technology represents a significant challenge to reverse engineering. New computational methods are required to reason quickly and effectively about programs that have been virtualized.

2009 Accomplishments

Our team developed new tools for reverse engineering virtual machine protected malware. These tools were successfully applied to an existing sample of virtualized malware yielding a new, decoded sample that was then more easily analyzed using traditional techniques.

2010 Plans

Part of the virtualization effort involved creating specialized algorithms for *code generation*. That is, given a sequence of instructions, generate a new, possibly different sequence of instructions that accomplishes the same functional effect but in a simpler or more direct fashion. While these algorithms were not part of, nor planned for, the existing FX system, they have proven very useful. We plan to extend the existing FX system to the more general problem of code generation by "backporting" and generalizing the new algorithms developed to address malware virtualization.

We believe code generation technology can be applied not only to virtualization, but also to the more traditional polymorphism and metamorphism malware obfuscation techniques.

References

[1] R. Linger, M. Pleszkoch, L. Burns, A. Hevner, and G. Walton, "Next-Generation Software Engineering: Function Extraction for Computation of Software Behavior," in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, Kona, Hawaii, January 2007.

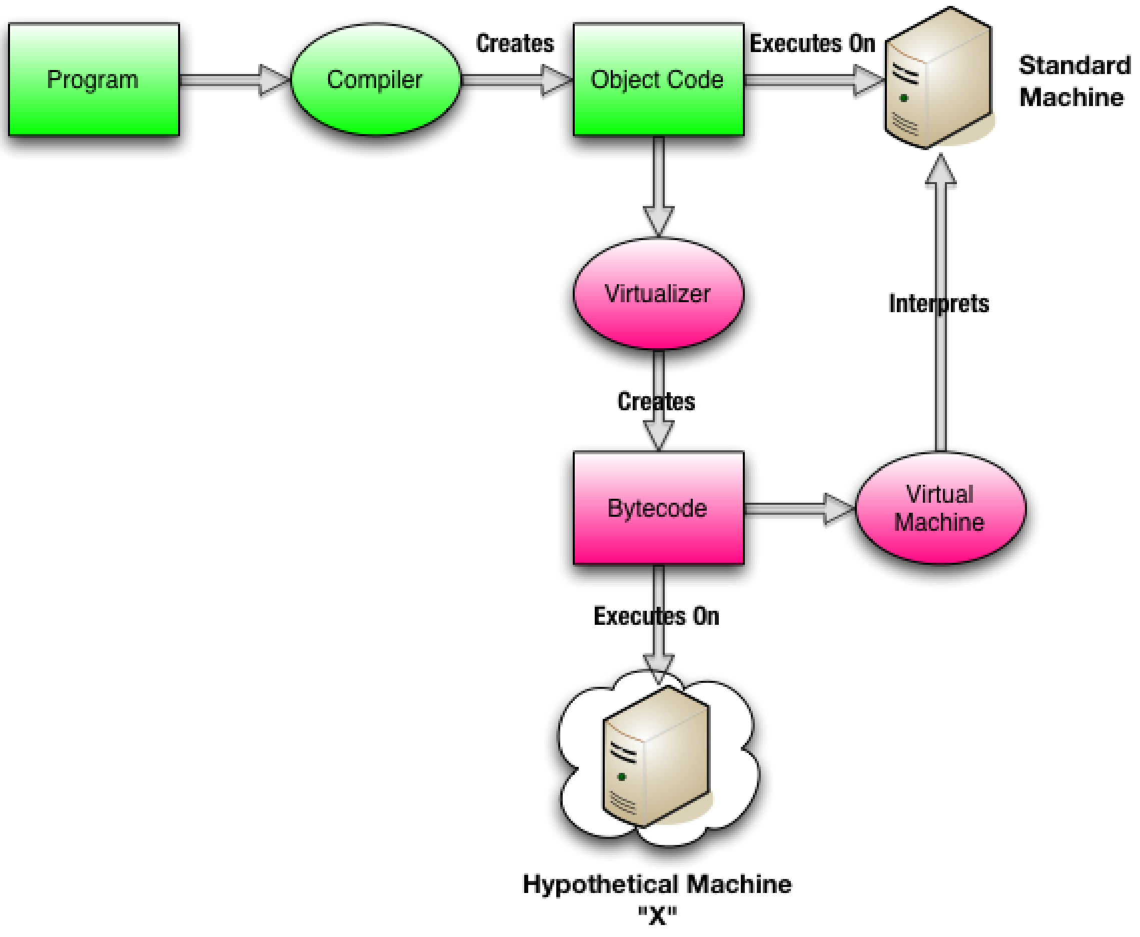


Figure 1: The Virtualization Process

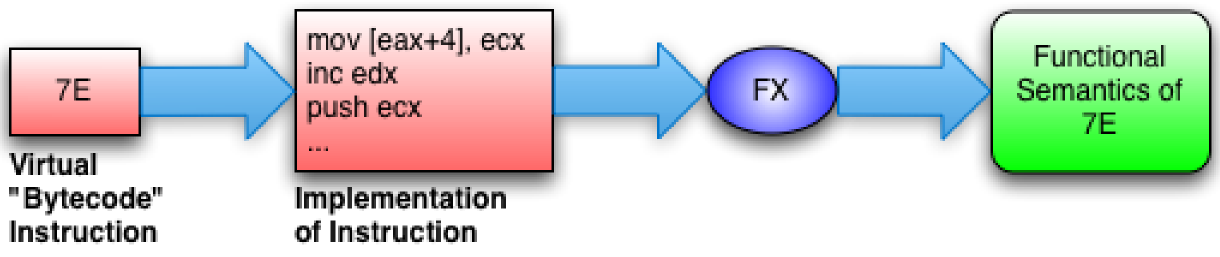

7E
Virtual "Bytecode" Instruction
mov [eax+4], ecx
inc edx
push ecx
...
Implementation of Instruction
FX
Functional Semantics of 7E

Figure 2: Understanding a Single Virtual Instruction

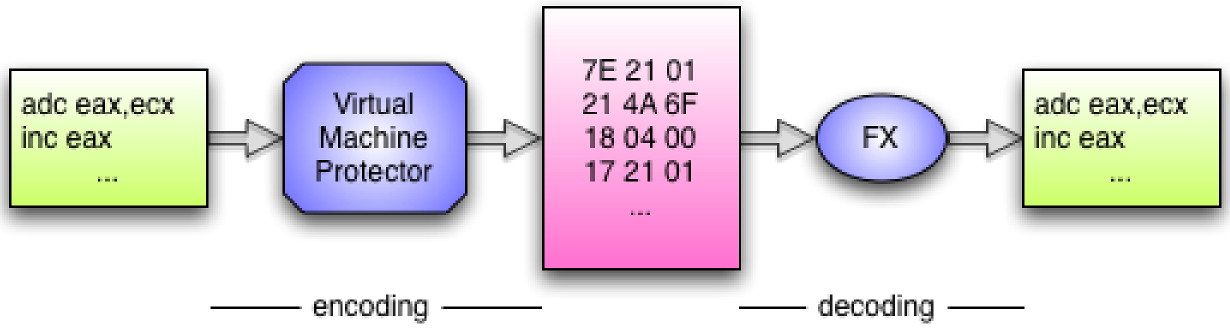
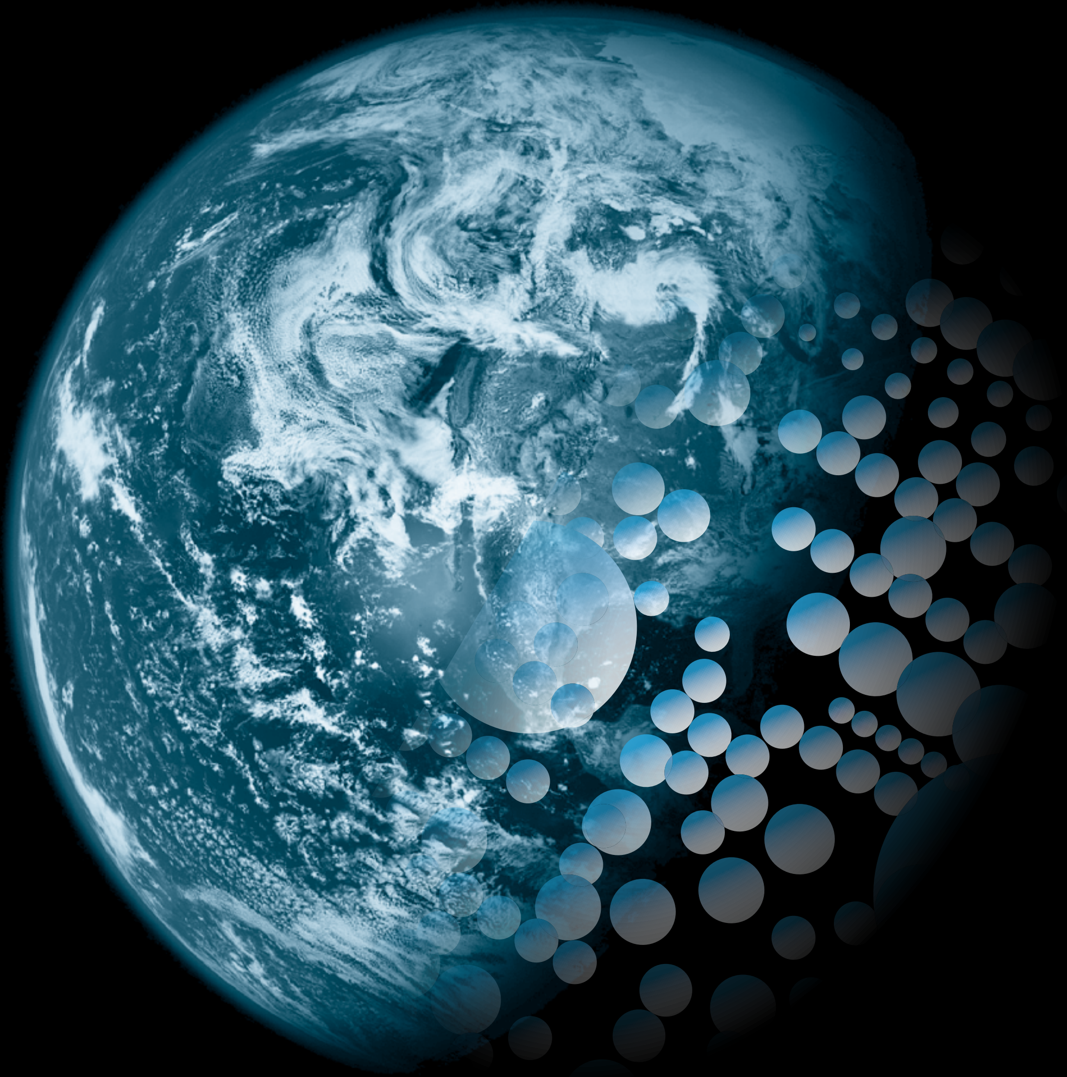


Figure 3: Encoding and Decoding the Original Program

Rhiannon Weaver

A Probabilistic Population Study of the Conficker-C Botnet



A Probabilistic Population Study of the Conficker-C Botnet

Abstract

Botnet size is often reported as a number of IP addresses, but the link between IP addresses and infected machines is more complicated than a simple one-to-one relationship. To count the number of infected machines when we have only an aggregated view of a botnet, we suggest building a precise probability model of the observable behavior of a single machine, and applying that model to the aggregates to obtain a population estimate. As an example, we build a probability model of the peer-to-peer (P2P) scanning activity of the Conficker-C botnet, and we use this model to estimate the number of active infected machines per hour over a two-month window.

Problem Addressed

When new botnets emerge, the classic question is, "How big is it?" In practice, population estimates are often made public in online articles or security blogs, where it is difficult to determine the methods used to obtain either the estimate or the margin of error. Population estimation has a long history in the statistical literature, but methods often involve complex models that track each individual unit in the population over time. These methods are difficult to implement over millions of individuals on an Internet-wide scale, and they also assume an observer has a direct view of the units of interest– individual machines. Often the view of infected machines in a botnet is filtered through IP space. The existence of Network Address Translation (NAT), proxies and Dynamic Host Configuration Protocol (DHCP) leases in IPv4 space complicates the link between IP addresses and machines. For example, if 100 infected machines are scanning a network from behind a NAT gateway, the outside observer sees only the aggregate scan attempts from all 100 hosts, coming from the single gateway IP address.

In this research, we consider a principled but computationally approachable model for measuring the number of active hosts per hour in a botnet, based on modeling activity for a single host. We apply our method to the Conficker-C botnet that emerged in March of 2009.

Research Approach

Our method relies on precisely describing a probability distribution for the observable behavior of a single infected host, and using the expected value of the distribution to represent a single host when we observe only aggregated measurements. In this context, "observable behavior" is a quantitative measurement that we can see with a network telescope, for example scanning rates, beaconing frequencies, or a count of remote file downloads. The behavioral model should account for network variability, empirical testing results, and any stochastic elements in either the underlying protocol or measurement method. It can be informed by sandbox experiments on the malware associated with a botnet, or from reverse engineering of the botnet's source code.

Once developed, the model provides a theoretical average value for the measured behavior across all hosts, as well as a measure of variance. Given an aggregated count, for example a stream of scan attempts coming from a gateway IP address, we divide the observed value by the theoretical average in order to obtain the population estimate. Statistical theory [1] proves that this estimate is unbiased for the true population when the model is precise and the theoretical average is accurate. We can also obtain a margin of error for the population size that is informed by the variance of the single-host probability distribution.

Expected Benefits

As Internet addressing moves away from the single-host, single-IP model, for example with the highly ephemeral nature of IPv6 addresses, behavioral models based on single-host probability distributions will still be able to provide measurements and metrics in terms of counts of machines. This ability allows for more interpretable and concrete results when we discuss the size of a population. Furthermore, not only can this methodology be used to track the size of homogenous populations such as botnets, but it can also be used to build inventory models for network situational awareness when the network monitoring agencies have either limited sensor placement, or only a high-level view of a very large network.

2008 Accomplishments

In 2008, we used the single-host probability model to study the Conficker botnet. The Conficker-C worm variant that propagated through hosts infected with Conficker-A and Conficker-B in March 2009 introduced a specific pattern of peer-to-peer (P2P) activity using both TCP and UDP protocols. When a host infected with Conficker-C comes online, it searches for peers by randomly generating a set of destination IP addresses across most of IPv4 space, attempting connections to these hosts, and adding any peers it encounters to an internal list. Connection ports are based on a deterministic algorithm that uses the source IP address and date. This algorithm was the key to a behavioral signature for the identification of Conficker-C P2P traffic with high reliability that can be observed in the large-scale summary information contained in network flow data.

We used the behavioral signature to obtain counts of UDP scan attempts from external Conficker-infected IP addresses into our network of approximately 21,000 class C net blocks (.15% of IPv4 space), recorded hourly from the period of March 5th through April 24th, 2009. Overall, we observed 38 million unique IP addresses during the 2-month period, across 1.09 million class C net blocks. Figure 1 displays the distribution of the average scan attempts per hour by class C net block using a log-log plot. While the majority of net blocks scanned our network between 3 and 5 times per hour when active, NATs and gateways show up as a long tail in the distribution, with some blocks scanning our network an average of over 1000 times per hour.

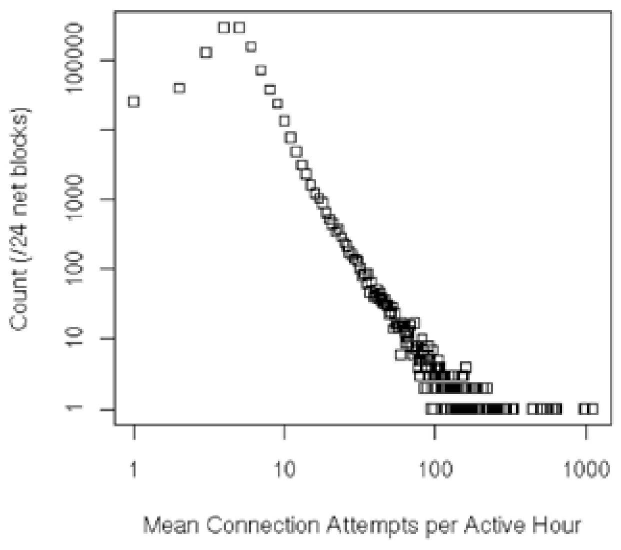


Figure 1: Variation in scan volume across Class C (/24) net blocks

To determine an activity profile by hour in terms of infected hosts, we used a sandbox study from SRI International [2] and a reverse-engineered image of the Conficker-C P2P source code [3] to develop a model for the number of times a single, active infected host would scan our network in the duration of one hour. The sandbox study indicated that hosts tend to scan rapidly when they first come online (to attempts per minute interval), with rates declining steadily over the first 2 hours toward approximately attempts per minute interval. We modeled this time dependency by specifying three states an infected host could occupy during an hour in which it is actively scanning

1. "Start-Up": The host comes online and initiates P2P scanning in this hour. This state is characterized by a high scan rate per minute and a small peer list, with activity commencing at some point within the hour.
2. "Running": The host has initiated startup and is actively scanning for the entire hour. This state is characterized by a low scan rate per minute and a large peer list, with sustained activity throughout the hour.
3. "Shut-Down": The host has been actively scanning and goes offline at this hour. This state is characterized by a low scan rate per minute and a large peer list, with activity terminating at some point within the hour.

We assigned each external net block to a time zone based on its associated country code, and used a random sample of low-volume net blocks to estimate the probability that a host would be in each of the three states for each relative hour of the day, given that it is actively scanning in that hour. Figure 2 shows the results; the probability of Start-Up is highest at 7 a.m., while the probability of Shut-Down is highest during the late evening and early morning hours.

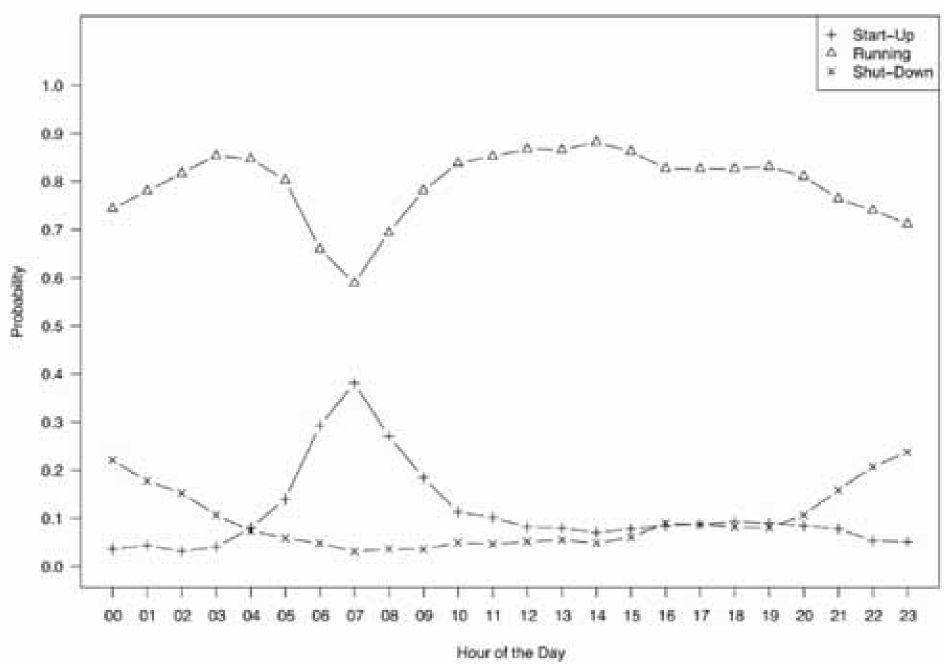


Figure 2: Probability of states (Start-Up, Running, and Shut-Down) by hour of day

Informed by the Conficker-C P2P protocol, we built Poisson probability models for the number of connection attempts we would expect to see in each state (Start-Up, Running, and Shut-Down). We then used a Binomial probability model to describe the distribution of the number of connection attempts directed to our network in each state. We averaged each of these models across the three states to obtain an overall model of observed scan attempts for each hour of the day. We used the theoretical average from this model to obtain population estimates.

Figure 3 shows the estimate of active Conficker-C hosts per hour for the two-month span starting on March 5th, and ending April 24th. The large jump occurs on March 17th and corresponds to a binary update that was released into the Conficker-C botnet. The largest host count associated with the botnet is 1.06 million active hosts. Numbers decline steadily through the month of April, but appear to stabilize toward the end of the month. The heavy lines correspond to a smoothed plot of both host count estimates (solid line), and observed unique IP address counts (dotted line). These lines show a trend that, as the botnet ages, it "spreads out" among IP space. The ratio of host count to IP count is large prior to the update in mid-March, but declines steadily afterwards. This decline in rate makes sense as large infected networks clean up. It suggests that the persistent infections of Conficker-C are among more isolated machines in IPv4 space.

2010 Plans

In 2010 we plan to compare the Conficker-C UDP scanning model to a model for Conficker-C's TCP scanning mechanism. We also hope to find a simplified method to account for "churn" in the botnet; currently the single-host model can estimate active hosts per hour, but not an overall number of hosts in the botnet, as it does not track individuals from hour to hour. We also hope to apply the basic methodology to the more general task of counting the number of hosts behind a gateway.

References

[1] Casella, G. and Berger, R. *Statistical Inference*. Duxbury Press, 1990.

[2] Porras, P., Saidi, H., and Yegneswaran, V. *Conficker C Analysis.* Technical report, SRI International, March 2009. http://mtc.sri.com/Conficker/addendumC/index.html

[3] Porras, P., Saidi, H., and Yegneswaran, V. *Conficker C P2P Protocol and Implementation.* Technical report, SRI International, September 2009. http://mtc.sri.com/Conficker/P2P/index.html

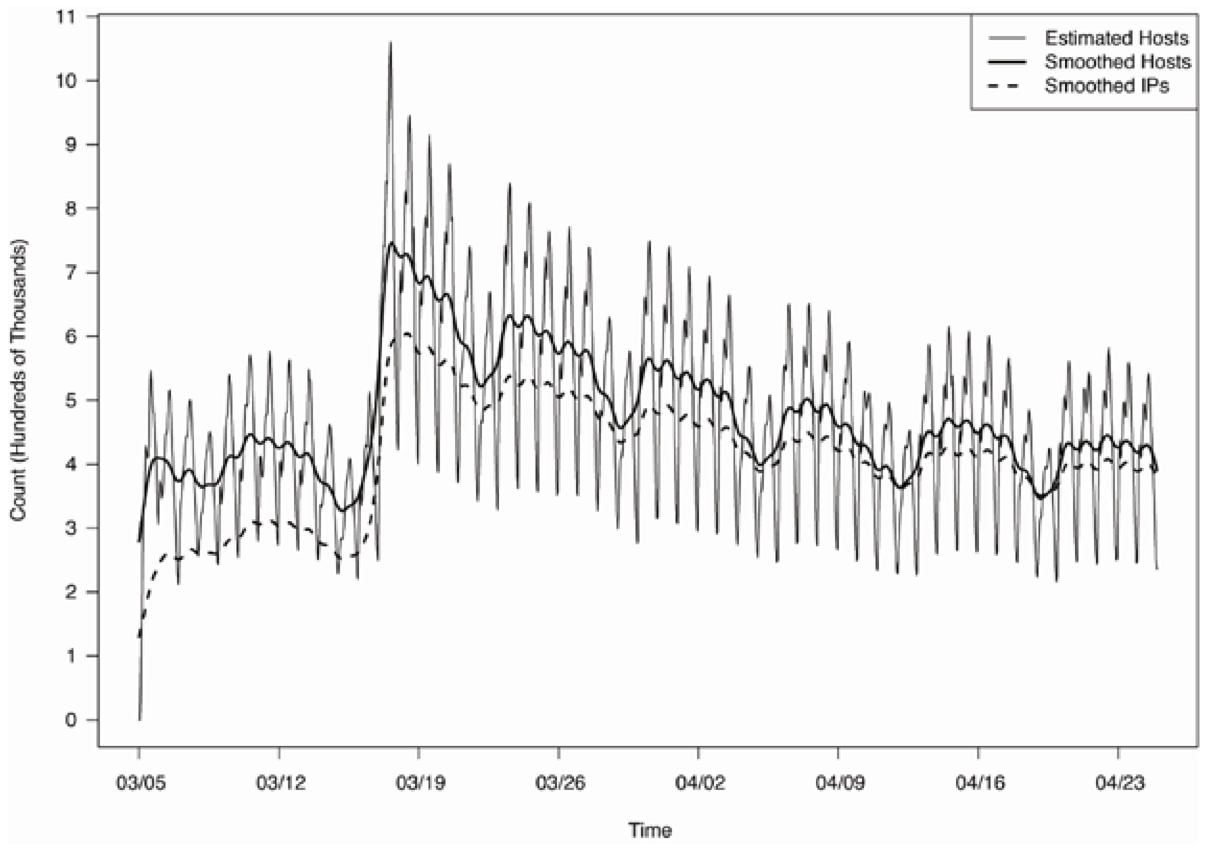
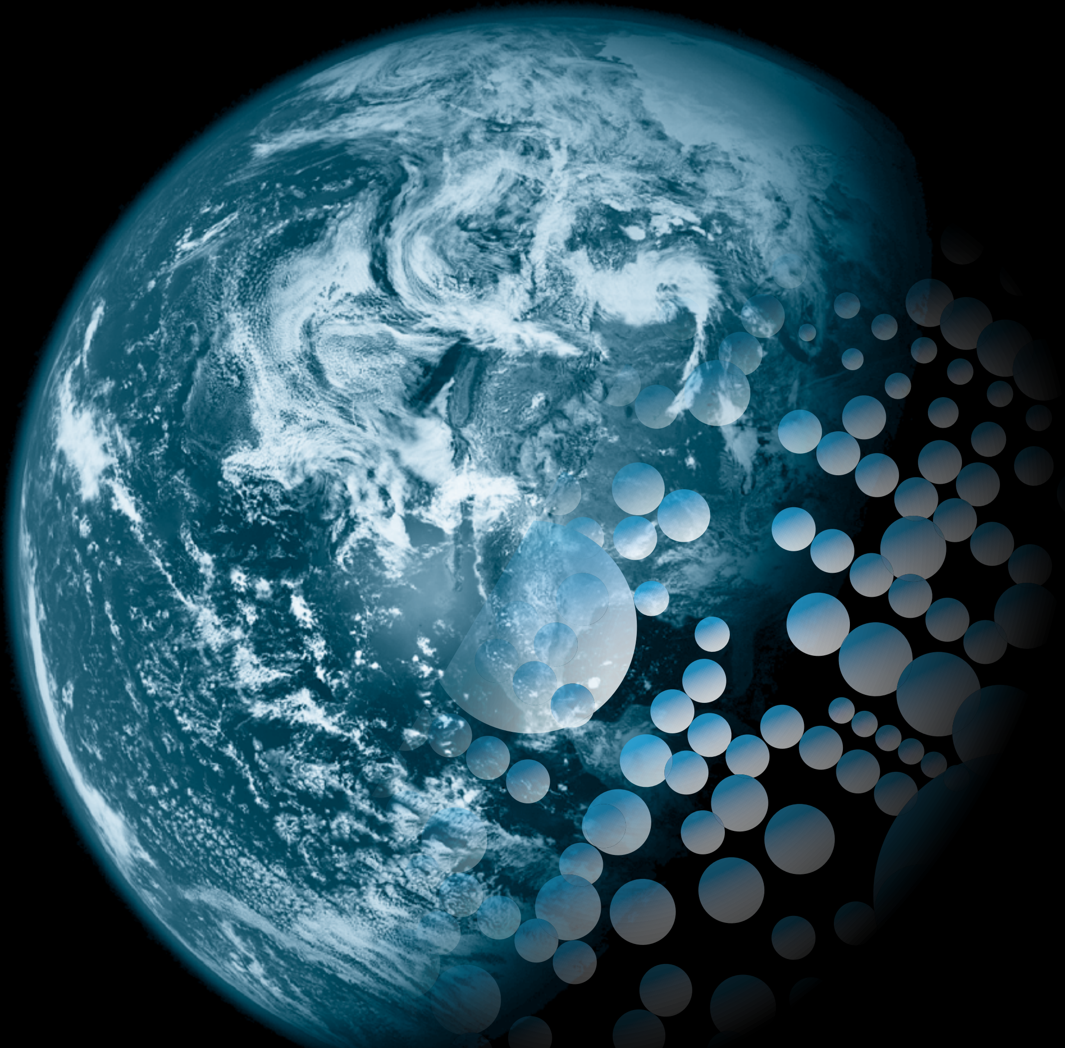


Figure 3: Estimated Host counts over time

Evan Wright

Catching IPv6 Tunneled in IPv4



Catching IPv6 Tunneled in IPv4

Introduction

Internet Protocol version 6 (IPv6) is the long-term solution for the depletion of IPv4 addresses and has been increasing in deployment over the past decade. To ease the adoption of IPv6, automatic tunneling protocols (i.e. 6to4, ISATAP, Teredo) will encapsulate IPv6 information inside IPv4 with minimal user effort. These protocols allow for IPv6 to run in networks regardless of the preparedness or awareness of the network administrators. In this work, we use features of these tunneling protocols to identify IPv6 traffic that is masquerading as IPv4.

Problem Addressed

The Internet Protocol (IP) is the fundamental protocol underlying the Internet, and it is in the process of an upgrade from version 4 (v4) to version 6 (v6). Since islands of IPv6 networks will likely exist while the majority of the Internet is IPv4, there is a need for a mechanism that enables communication. Most configurations of IPv6 hosts also run IPv4.

A tunnel allows for one protocol to be encapsulated within another. Figure 1 illustrates that IPv6 traffic gets wrapped in IPv4. If a sensor is not aware that some data packets are tunneled, it will mistakenly decode tunneled IPv6 as IPv4 traffic, with the real IPv6 data packet showing up as additional payload. Often, network flow sensors do not decode tunneled IPv6 traffic, yet network security and network situational awareness operations both have a need to be able to identify these tunnels in traffic.

Tunnel connection establishment can be either automatic or static. Automatic tunnels require one end of the connection to be specified and fixed; the other end does not need to be specified and may be in any location. Examples of automatic tunneling protocols include Teredo, 6to4, and the Intra-Site Automatic Tunneling Addressing Protocol (ISATAP). Static tunnels require configuration at each end, and each endpoint must be static.

IPv6 can negatively affect security. Firewalls and Intrusion Detection Systems (IDSs) inspect native and tunneled IPv6 traffic for two reasons: to ensure tunneled IPv6 does not bypass security measures, and for future native IPv6 deployment. Once identified, IPv6 tunnels should be either verified and then hardened or entirely blocked.

Tunneling behavior is expected to increase before IPv6 is used natively on a wide scale for two reasons. First, Microsoft and other vendors are releasing applications with greater use of IPv6 then they had released previously. Second, operating systems, such as Windows, are being released with greater reliance on IPv6 support.

Research Approach

Based on the unique network behavior, the tunneling protocols are grouped into three categories: Teredo, 6to4 and other IPv6 tunnel traffic. Teredo and 6to4 are both automatic tunneling protocols. The other IPv6 tunnel traffic category includes statically configured tunnels and the ISATAP automatic tunneling protocol. For a security analysis, it can be very difficult to determine which machines are using automatic tunneling protocols [1]. Netflow analysis allows for easy identification of automatic IPv6 tunnels, in a scalable manner. Flow analysis provides a historical dimension to our data, allowing insight into relevant past behavior that isn't available with an instantaneous full packet view. PySiLK [2, 3] is an extension to normal flow analysis that provides the capabilities to track states of a protocol. We used PySiLK to create scripts that identify these IPv6 tunnels.

Both Teredo and 6to4 have unique distinguishable characteristics that separate them from other IPv6 tunneled traffic. We leverage these differences to distinguish Teredo from 6to4 and to distinguish both from all other IPv6 tunneled IPv6 traffic. Teredo contacts a short list of possible servers for tunnel connection establishment via User Datagram Protocol (UDP) port 3544 and proceeds to connect to the Teredo relay for the rest of the connection.

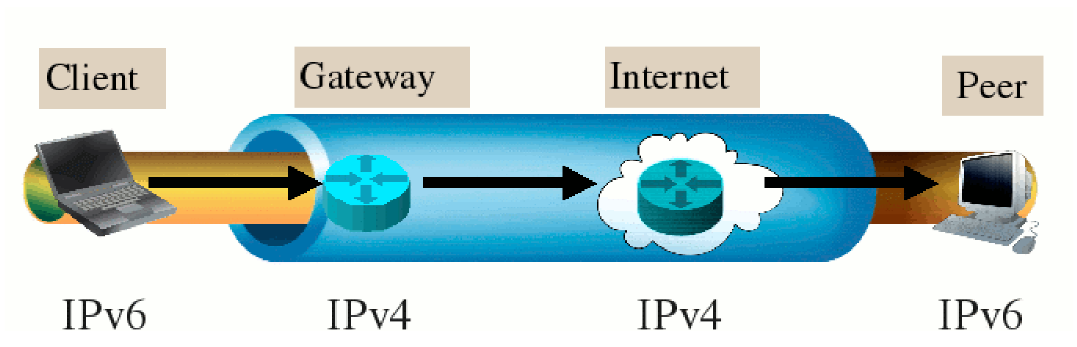


Figure 1: IPv6 in IPv4 Tunneling

Our scripts use PySiLK to maintain state across flows for the Teredo and 6to4 protocol identification. First, IP addresses that have contacted the Teredo server IP addresses are noted. Then all the subsequent traffic from these IP addresses is recorded. Finally, other traffic from other erroneously identified traffic is filtered out. It is necessary to filter out other traffic because it could be possible for a Teredo host to communicate to Teredo servers then send other UDP port 3544 traffic that is not Teredo to another destination. 6To4 uses protocol number 41, in contrast to UDP, and establishes the tunnel with an IP at a specific anycast address (192.88.99.1). Similar to Teredo, we track the state of connections that have sent a unicast query. Protocol 41 traffic is very low volume and not affected by the noise of UDP communication. Subsequent activity can be with any arbitrary IP address over protocol 41. All protocol 41 traffic that does not match a previous signature will be a manually established tunnel or ISATAP.

Expected Benefits

IPv6 in IPv4 tunneling is widely supported by many operating systems, including Windows XP SP3, Windows Server 2008, and all newer versions of Microsoft desktop and server operating systems. IPv6 tunnel detection makes security administrators aware of the prevalence of IPv6 tunneling, so that action can be taken and the overall security improved. Often, administrators assume that since native IPv6 is blocked that IPv6 using tunnels could not be running on the network and on hosts in the network. Network defenses, such as packet filtering, inspection, and alerting, may not be prepared for such tunneling in unprepared networks.

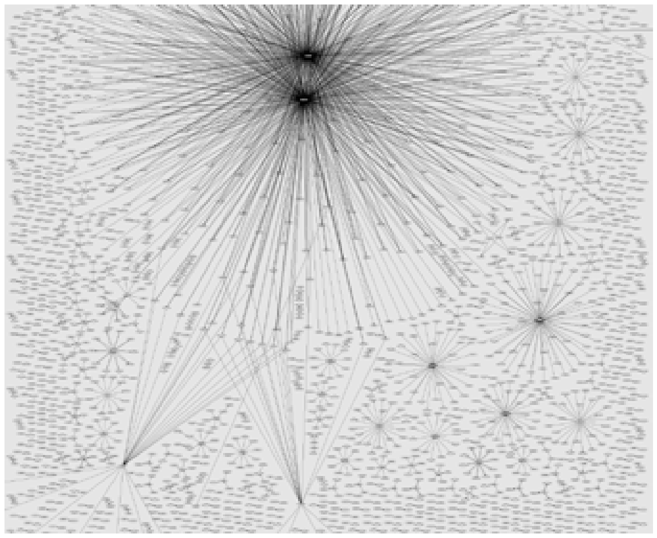
Despite the security risks of IPv6 tunnels operating on a network, IPv6 enables many benefits. By identifying the IPv6 tunnels, the administrators have the option to transition into an IPv6 network in a more gradual way without detrimentally affecting security. Some benefits of IPv6 include increased address size, faster switching of packets, and increased scalability of routing tables.

2009 Accomplishments

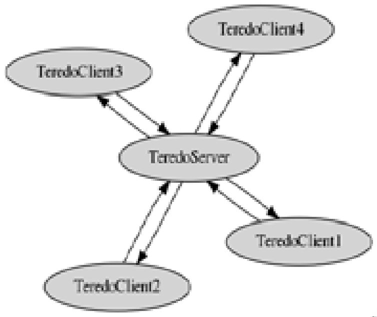
In 2009, we implemented a system for detecting IPv6 tunneling and deployed it over a large network. This system can detect IPv6 tunnels operating, even when the equipment on the large network cannot process IPv6. IPv6 tunnels are rarely known to administration. Often, the IPv6 tunnels may be a channel for malicious activity.

The IP addresses communicating to servers and traffic volumes and timing information about devices using IPv6 tunneling can be used by security administrators. The security administrators may also identify specific IP addresses doing this IPv6 tunneling and disable them. In practice, IPv6 tunneled traffic exists on large networks often even if it is prohibited. The volume and interactions of IPv6 tunnels can be measured on a large scale. More specifically, the data from this system can be visualized over time with a time-series plot.

Another way to use this data is to visualize the pairwise interactions of IP addresses communicating over tunneled IPv6 in a given time period. Figure 2a shows traffic IP address pairs communicating with one IPv6 tunneling protocol, Teredo, in a large network over a three-month period in 2008. The two highly connected vertexes are two of the Teredo servers hosted by Microsoft. Many smaller star-type clusters appear that are Teredo servers, if they have bidirectional traffic. Figure 2b shows a simplified example of the interactions the bidirectional traffic. A unidirectional traffic example is in Figure 2c. The star-type clusters that are unidirectional are scanners, misconfigured Teredo clients, or misconfigured Teredo servers.



(a)



(b)

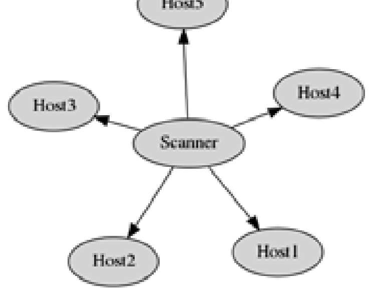


Figure 2: IP Address Interactions of Teredo Traffic

(c)

2010 Plans

In 2010, we will investigate how we can fully automate detection of the IPv6 tunneling protocols that are already identified. Existing Teredo identification is done by correlating known, existing Teredo servers with traffic patterns. In the future, we will attempt to statistically identify the patterns of the Teredo servers themselves so that new Teredo servers can be discovered automatically.

Another future goal is to increase the number of tunneling protocols that are identified. The ISATAP protocol, in particular, may have some characteristics that can be identified with a more in-depth machine learning approach, similar to [4]. In rare cases of 6to4 traffic, misconfigured Teredo servers also may not be properly identified, and a statistical analysis may be able to improve these rare cases.

References

[1]Hogg, S., Vyncke, E. “IPv6 Security.” Cisco Press, 2009 (ISBN 978-1-58705-594-2) p. 483.

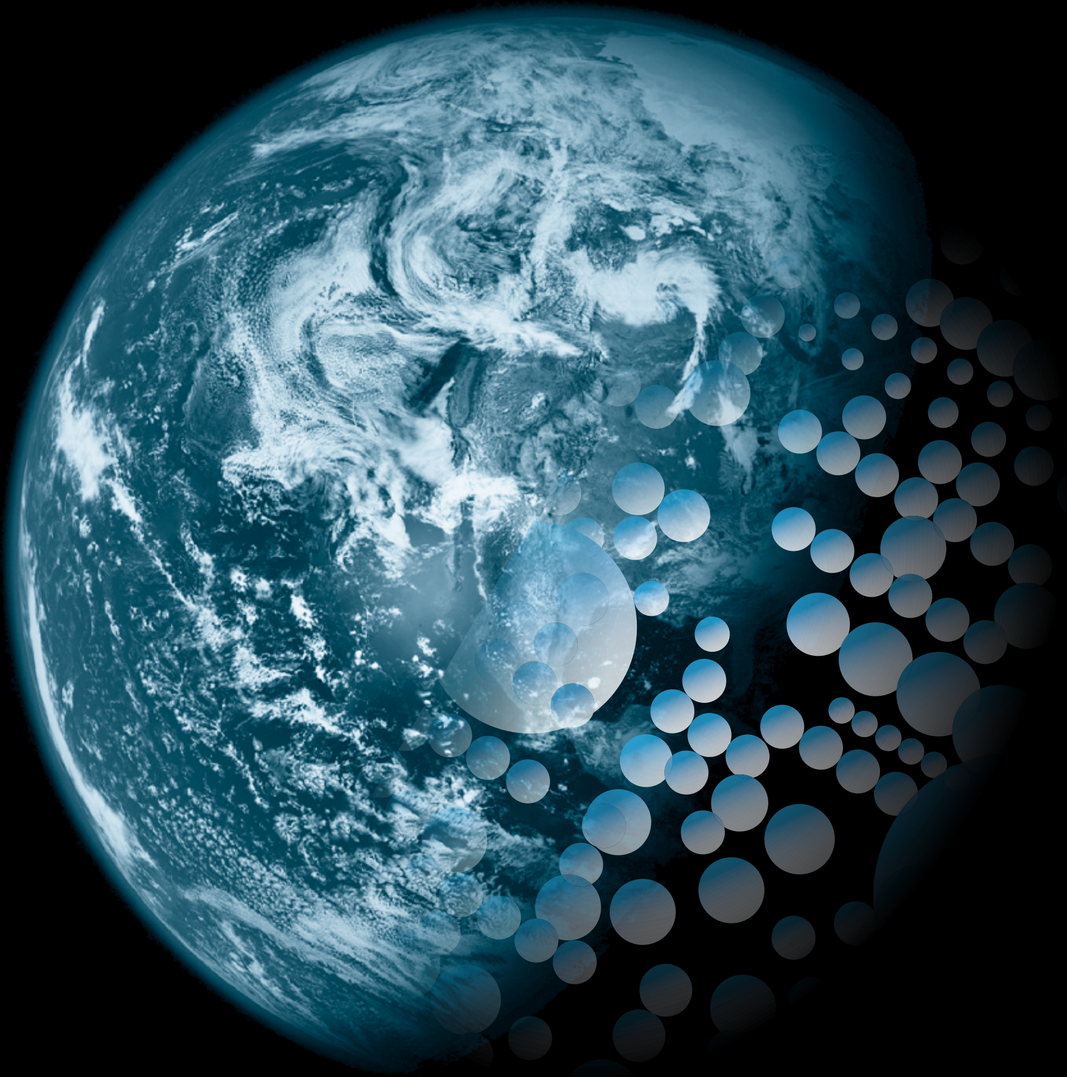
[2]Duggan, M, Shimeall, T. “PySiLK - A Language for Scripted Flow Manipulation” 2008 CERT Research Annual Report.

[3] PySiLK: SiLK in Python. http://tools.cert.org/silk/silkpython.html.

[4]Duffield, N., Haffner, P., Krishnamurthy, B., Ringberd, H. “Rule-Based Anomaly Detection on IP Flows.” *IEEE Infocom 2009*. http://www2.research.att.com/~duffield/papers/snort_infocom.pdf.

Ed Stoner

Finding Malicious Activity in Bulk DNS Data



Finding Malicious Activity in Bulk DNS Data

Abstract

The Domain Name System is a vital component of the Internet, and nearly every transaction on the Internet uses it. It contains a wealth of Network Situational Awareness information that can be used to discover malicious traffic. This report describes specific techniques to detect certain types of malicious traffic. These techniques have been developed through analyzing a large amount of DNS traffic data. CERT has developed specific tools that apply these techniques in an ongoing way. Future research will include enhancing the developed tools, developing new techniques and tools to work with known malicious patterns, and discovering new malicious patterns.

Problem Addressed

The Domain Name System (DNS), which maps names to IP addresses, is a vital component of the Internet. Nearly every transaction on the Internet begins by making a DNS query. This is true for both benign and malicious activity. Having access to large amount of DNS queries allows us to look for patterns in Internet transactions. In certain cases, the patterns for malicious activity can be distinguished and identified.

Research Approach

The Security Information Exchange (SIE) is a framework for information sharing run by the Internet Systems Consortium (ISC). It gives researchers access to a large amount of DNS messages. CERT has a server setup at SIE that captures information for about 400 million DNS messages per day. DNS messages contain either a question about a host name (i.e. what is the IP address of www.google.com?) or an answer to a question. The DNS messages that are streamed by SIE are all answer messages that are marked as authoritative and do not contain errors. These messages are streamed to the CERT server and stored in the ncap file format developed by ISC specifically for this type of application. The ISC provides a piece of software called ncaptool to send, capture, and process these streams of DNS messages. With ncaptool and additional software developed at CERT, we are able to analyze the stored DNS messages for particular patterns of malicious activity.

Some of these patterns are the result of using the DNS infrastructure itself in a way that it was not originally designed for. A particular example of this is DNS tunneling. Other patterns of malicious activity are detectable because of certain constraints that the type of malicious activity places on how DNS can be used. An example of this is Fast Flux hosting.

DNS tunneling is a process where DNS messages are used to transport arbitrary data by encoding that data into the DNS messages themselves. Because of the very wide support and availability of the global DNS infrastructure, and because very few organizations block DNS traffic from individual clients to the Internet, this method can be very effective for bypassing security measures such are firewalls or ACLs.

A DNS tunneling implementation is detectable when it is created to encode arbitrary data, and is either used for two-way communication or data exfiltration. The reason for this is that in a DNS question (which is what a client would use to pass information outbound), the only place to encode information is in the host name. Per the DNS protocol specification (RFC 1035), the host name has only 63 allowable characters (all upper and lower case letters, digits 0 through 9, and hyphen). In order to encode arbitrary data and achieve reasonable bandwidth, implementations of this type of tunneling will use noticeably more unique characters than normal host name would have. Figure 1 shows the unique character counts for a host name used in DNS tunneling and for www.google.com.

host name	unique characters
08f0b06a25a5cf1f9df501bc39306 fbc6ff7875646817b4845c17da0.6. ewsxz.com	23
www.google.com	8

Figure 1: Unique character count of host names

It is also a feature of the DNS protocol that the question is contained in the answer message. So even though our data only has DNS answers, it is still possible to find the tunneling. This is done by iterating through a particular set of messages, extracting the host name in the question, and counting how many unique characters are in it. After some initial testing it was determined that 20 unique characters was a good starting point for finding DNS messages with encoded information.

In fast flux hosting, an attacker uses DNS to hide malicious sites behind an ever-changing network of compromised hosts. This pattern shows up in DNS records as an unusually large number of distinct IP addresses in answers returned for the query of a single domain name, with each answer having a very short period of validity or time to live (TTL), and with previously unseen IP addresses constantly emerging in queries over time. The answers given all point to a proxy network—a set of compromised machines that relay traffic to a central host or small set of hosts that the malicious party controls. This proxy network hides the real malicious site, making it difficult to track the site and to take it down.

To find Fast Flux hosting in our data, we first collect answers which give more than ten IP addresses for a given host name and have a TTL of 2000 seconds or less. Next, we count how many unique IP addresses are seen for each host name in the data we collected. Lastly, for host names with more than 25 unique IP addresses, we count how many different ASNs (Autonomous System Numbers, which map to Internet Service Providers) there are. If there are more than 20, then it is extremely likely that Fast Flux hosting is happening for the host name.

Answering with multiple IP addresses for a question about a host name is a long-standing legitimate practice to provide redundancy and high-availability. What makes Fast Flux different and detectable is that the hosts are compromised. Therefore, unlike a legitimate service, they are much more unreliable, so more hosts are needed, and they need to have a shorter TTL. Secondly, the hosts need to be on very diverse networks, or otherwise it would be easier to shut them off. Taking advantage of these characteristics yields a very successful algorithm.

In researching Fast Flux hosting, we noticed a few other characteristics that are prevalent in domains setup for malicious activity. One is that host names have a computer generated label. The other is that a top-level domain (like .com or .net) will appear in the middle of the name (www.somebank.com.badguy.tv). We can use these characteristics to sort out malicious domains from compromised domains in existing block lists (which can then be used to track down hosts that are visiting malicious domains).

Expected Benefits

By identifying malicious activity in DNS messages, we can begin to develop a more proactive monitoring approach. Rather than relying on hand-assembled lists of malicious activity, we can start automatically generating lists that have the potential to include zero-day and previously unnoticed attacks and attack vectors. This can provide for quicker incident response time and the ability to notice a wider range of incidents.

2009 Accomplishments

The tools to apply the Fast Flux detection algorithm described to large amounts of DNS data have been developed. A system has been put in place to find Fast Flux domains as well as hosts on certain networks that are connecting to those domains.

The tools to find DNS tunneling and exfiltration have been developed. Because there are so many examples of legitimate exfiltration, the techniques are very sensitive as to what network they are monitoring.

The tools to separate malicious domains from known lists of bad domains and find the IP addresses associated with those domains have been developed. These IP addresses can then be used by systems already in place that monitor activity by IP address.

2010 Plans

One challenge that came up in our research into finding DNS tunneling was determining which tunneling messages are malicious. Because of its effectiveness and wide support, DNS tunneling has been adopted by many organizations for providing Real-time Blackhole Lists and similar lookup services. Future work will include developing techniques to identify whether a tunnel is benign or malicious.

The current tools for distinguishing between malicious domains and benign domains are very crude, and simply check for a large number of unique characters and top-level domains that are in the middle of a name. Because of this they require existing lists of known bad domains. We are exploring new techniques to be able to detect malicious domains with more reliability and without pre-filtering. That work includes identifying better algorithms to determine if a label in a DNS host name is computer generated.

By continuing to develop and refine our toolset for analyzing DNS messages, we hope to identify additional malicious characteristics, and to provide additional tools to detect and report incidents. By combining the DNS data with other data sources we hope to detect new types of malicious activity.

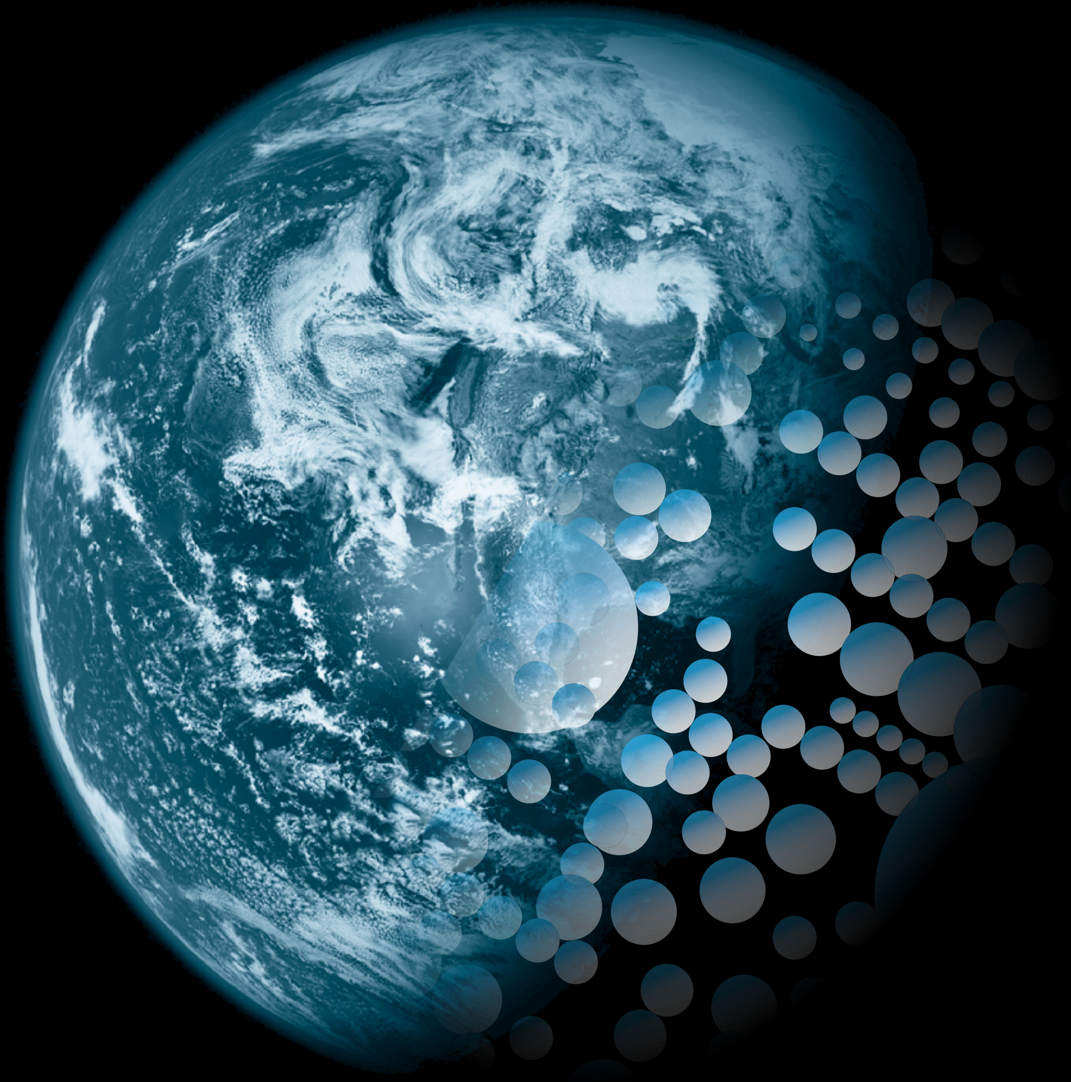
References

[1] Tactical IT. “DNS Traffic Analysis.” http://tactical-it.com/2009/01/a-study-of-dns/

[2] Thorsten Holz, Christian Gorecki, Konrad Rieck, & Felix C. Freiling, “Measuring and Detecting Fast-Flux Service Networks,” 2008 ISOC Network & Distributed System Security Symposium.

[3] Jose Nazario, Thorsten Holz, “As the Net Churns: Fast-Flux Botnet Observations,” 3rd International Malicious and Unwanted Software (Malware 2008).

Function Extraction for Malicious Code Analysis



Kirk Sayre

Mark Pleszkoch

Timothy Daly

Richard Linger

Stacy Prowell

Function Extraction for Malicious Code Analysis

Problem Addressed

As the quantity and sophistication of malicious code continues to grow, automation support for analysis becomes more important to keep pace with the scope and scale of the problem. To help address this need, CERT originated and continues to evolve the technology of Function Extraction (FX) for automated computation of software behavior (including malware) with mathematical precision to the maximum extent possible.

Research Approach

The objective of FX is to replace slow and fallible manual methods of code analysis with fast and correct computation of behavior. Computing the behavior of software requires deriving its net functional effect, that is, how it transforms inputs into outputs in all circumstances of use. That information can be presented to analysts in non-procedural behavior displays that define all the possible effects a program can have, essentially, an "all cases of behavior" view. The foundations for behavior computation are found in the mathematics of denotational semantics [1,2,3,4]. Individual instructions of a language are treated as rules for mathematical functions, that is, mappings from their domains to ranges, or inputs to outputs. These mappings are pre-defined as a starting point for the FX computational process, which composes the functions of instructions in an input program in a stepwise process to ultimately arrive at the net functional effect of the entire program [5]. Methods have been developed for loop behavior computation that can make the effects of theoretical limitations on this process arbitrarily small. The resulting computed behavior is expressed in the form of a conditional concurrent assignment (CCA). A CCA denotes a function from program input to program output, and is composed of a collection of predicates that partition the input space. For each predicate an associated concurrent assignment from the input state to the output state denotes the net functional effect implemented by the program when the predicate is true. The general behavior computation process is as follows:

- Functional transformation
  Transform input program instructions into pre-defined equivalent function mappings.
- Structuring transformation
  Transform the spaghetti logic of an input program into systematic structured form with any arbitrary jumps and branches eliminated, and expressed in a hierarchy of standard control structures, including sequence, ifthenelse, and whiledo.
- Behavior computation
  Compute the behavior of control structures in the hierarchy, starting at the leaf nodes and progressing through compositional operations up the tree to arrive at the net functional effect of the entire program.

CERT is developing an FX system that computes the behavior of programs written in or compiled into Intel assembly language. The Function Extraction for Malicious Code (FX/MC) system is a specialization of FX technology that focuses on malware, in particular, on elimination of certain forms of intentional code obfuscation that can make analysis more difficult. Intruders employ many methods to make their malware resistant to analysis and thereby more effective in carrying out their objectives. Common methods of obfuscation include

- Control flow obfuscation
  Intruders often obfuscate the control flow of malware packages using commonly available malware development tools to insert massive quantities of arbitrary jumps that thwart human efforts to trace the control flow of the malware code.
- No-op code insertion
  Intruders frequently insert sequences of "no-op" code (code with no persistent effect) in substantial quantities in malware packages. No-op code has the appearance of normal functional code, but has no functional effect. Because analysts have no *a priori* knowledge of what code is no-op and what is truly functional, much time and effort can be consumed in analyzing code that turns out to have no impact on the overall functional effect of the malware.

The FX/MC system addresses these problems as follows:

- Eliminating control flow obfuscation
  The structuring capabilities of the FX/MC system trace and eliminate obfuscated control flow produced by arbitrary jumps. This process creates a function-equivalent representation of the original program expressed in nested and sequenced single-entry, single-exit control structures, including sequence, ifthenelse, and whiledo. With obfuscated control flow eliminated, the malware is reduced in size and complexity, and analysts can more readily understand its structure and function.
- Detecting no-op code
  FX/MC computes whatever behavior is produced by the control structures of an input program. The behavior may turn out to have an actual functional effect, or may have no functional effect at all (no-op code). The system employs behavior computation to detect and eliminate no-op code, thereby freeing the analyst to concentrate on portions of the malware that produce actual functional effects.

In illustration of elimination of control flow obfuscation by FX/MC, consider the code of Figure 1. This display depicts part of a malware program that has undergone obfuscation by a popular intruder tool that inserts large numbers of arbitrary jumps as shown by the red arrows. The resulting complexity of control flow can exceed human comprehension. Automatic structuring by FX/MC eliminated the arbitrary jumps and reduced the malware to its original form, in this case a simple sequence structure of just six readily understandable instructions containing no branching or looping logic. The unobfuscated version of malware produced by F/MC is not depicted here for obvious reasons.

In illustration of no-op code detection, consider the display of Figure 2 that depicts FX/MC in operation. The top-left pane shows the build-up of a sequence of instructions that have been previously added one at a time from a malware program by the user. Each time an instruction is appended, the system computes the net behavior of the entire sequence. This computation automatically eliminates any detected no-op code, leaving only those instructions with actual functional effect, as shown in the top-right pane. The two listings are identical at this point, indicating that no no-op code has been detected. The bottom pane depicts the net computed behavior of the entire code sequence.

The top-left pane of Figure 3 shows addition of one more instruction to the sequence of Figure 1. Behavior computation now identifies a section of the sequence as a no-op, as indicated by the absence of several instructions with no functional effect in the top-right pane. This code need not be investigated by the analyst. This miniature example involves a few lines of code. In practice, no-op code blocks can be large and numerous, and their automatic elimination can save analysts substantial time and effort.

Expected Benefits

Automation support for elimination of control flow obfuscation from massive inclusion of arbitrary jumps and detection of no-op code sequences can help analysts to more quickly determine the functional intent of malware and develop countermeasures. As experience with the FX/MC system grows, additional uses for behavior computation in malware analysis will likely emerge. Beyond malware analysis, FX technology can be expected to help DoD with problems of malicious code detection [6], anti-tamper and corrupted function analysis, computational analysis of security attributes [7], correctness verification, validation of embedded systems [8], reduction in testing costs [9], legacy system understanding, and creation of assured software repositories. In terms of general software development and analysis, a controlled experiment quantified the positive effect of availability of computed behavior on programmer performance [10].

2009 Accomplishments

The initial version of the FX/MC system was developed and delivered in 2009.

2010 Plans

Substantial evolution of the FX/MC system is planned at sponsor request. Additional capabilities include processing of malware code in a variety of file formats, extended analysis of complex malware control flow as a precursor to structuring and behavior computation, and development of an API interface for integration of FX technology with existing suites of malware analysis tools.

References

[1] Allison, Lloyd, *A Practical Introduction to Denotational Semantics,* Cambridge Computer Science Texts 23, Cambridge University Press, 1986.

[2] Smullyan, Raymond, *Recursion Theory for Metamathematics,* Oxford Logic Guides 22, Oxford University Press, 1993.

[3] Prowell, S.; Trammell, C.; Linger, R.; & Poore, J. *Cleanroom Software Engineering: Technology and Practice.* Addison Wesley, 1999.

[4] Mills, H. & Linger, R. "Cleanroom Software Engineering." *Encyclopedia of Software Engineering, 2nd ed.* (J. Marciniak, ed.). John Wiley & Sons, 2002.

[5] Linger, R., Pleszkoch, M., Burns, L., Hevner, A., and Walton, G. (2007). "Next-Generation Software Engineering: Function Extraction for Computation of Software Behavior," *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS40),* Hawaii, IEEE Computer Society Press, Los Alamitos, CA.

[6] Pleszkoch, M. and Linger, R. "Improving Network System Security with Function Extraction Technology for Automated Calculation of Program Behavior." *Proceedings of the 37th Hawaii International Conference on System Sciences* (HICSS-37). Waikoloa, HI, Jan. 5-8, 2004. IEEE Computer Society Press, 2004.

[7] Walton, G., Longstaff, T, and Linger, R. (2006). *Technology Foundations for Computational Evaluation of Security Attributes,* Technical Report CMU/SEI-2006-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

[8] Bartholomew, R., Burns, L., Daly, T., Linger, R., and Prowell, S., "Function Extraction: Automated Behavior Computation for Aerospace Software Verification and Certification," *Proceedings of 2007 AIAA Aerospace Conference,* Monterey, CA, May, 2007, Vol. 3, pp. 2145-2153.

[9] Linger, R., Pleszkoch, M., and Hevner, R., "Introducing Function Extraction into Software Testing," *The Data Base for Advances in Information Systems: Special Issue on Software Systems Testing,* ACM SIGMIS, New York, NY, 2008.

[10] Collins, R.; Walton, G.; Hevner, A.; & Linger, R. *The CERT Function Extraction Experiment: Quantifying FX Impact on Software Comprehension and Verification* (CMU/SEI-2005-TN-047). Software Engineering Institute, Carnegie Mellon University, 2005.

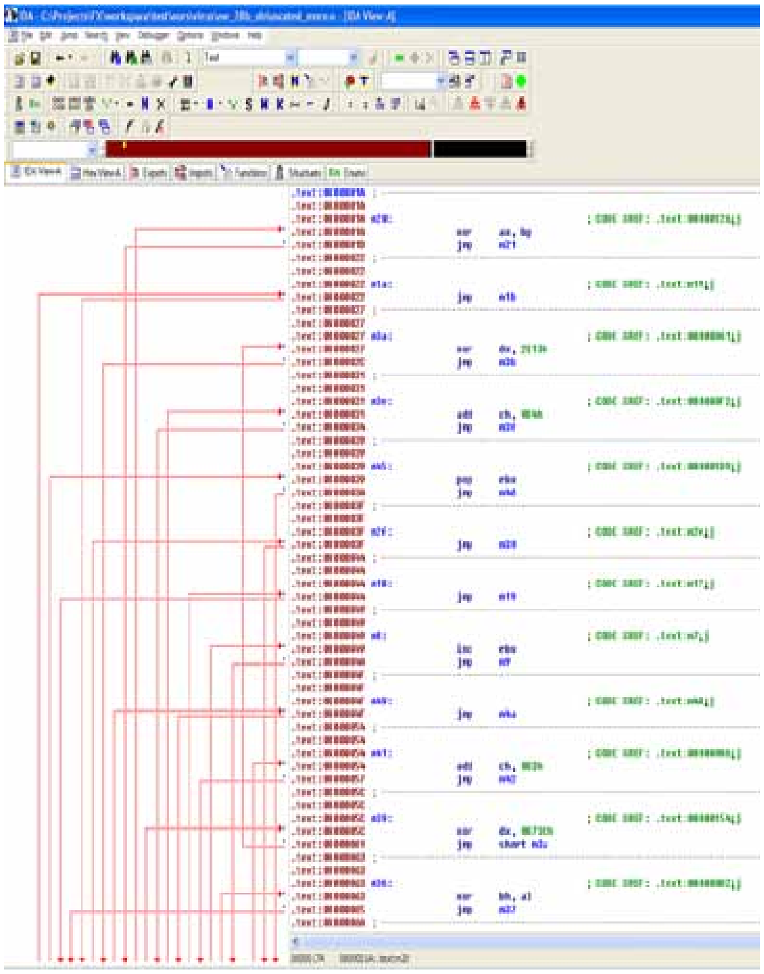
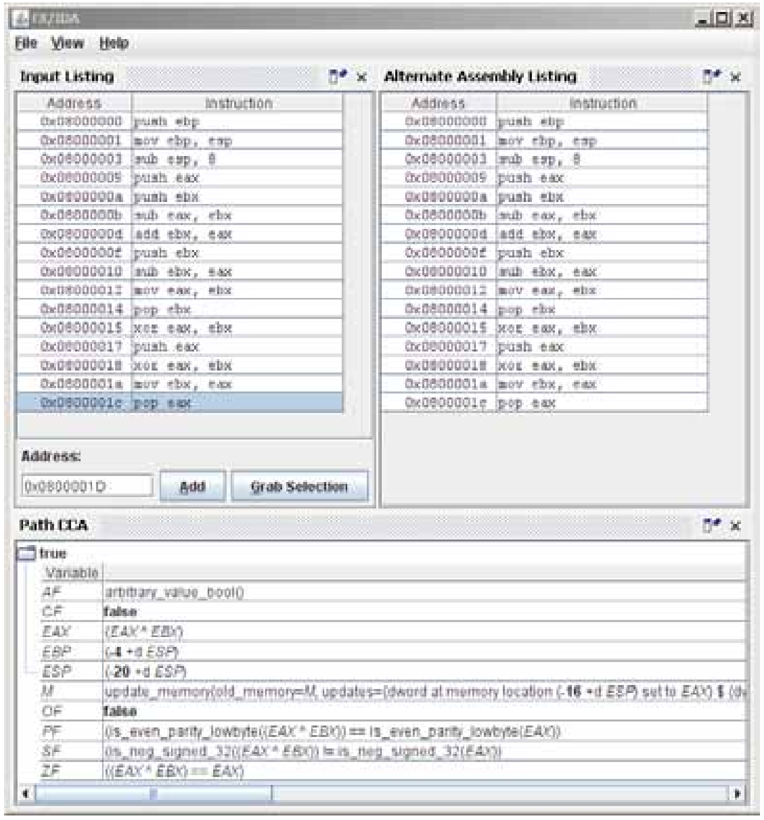


Figure 1: Malicious Code Exhibiting Control Flow Obfuscation For Elimination by FX/MC



Input Listing

Address	Instruction
0x08000000	push ebp
0x08000001	mov ebp, esp
0x08000003	sub esp, 8
0x08000005	push eax
0x0800000a	push ebx
0x0800000b	sub eax, ebx
0x0800000d	add ebx, eax
0x0800000f	push ebx
0x08000010	sub ebx, eax
0x08000012	mov eax, ebx
0x08000014	pop ebx
0x08000015	xor eax, ebx
0x08000017	push eax
0x08000018	xor eax, ebx
0x0800001a	mov ebx, eax
0x0800001c	pop eax

Address: 0x0800001D

Alternate Assembly Listing

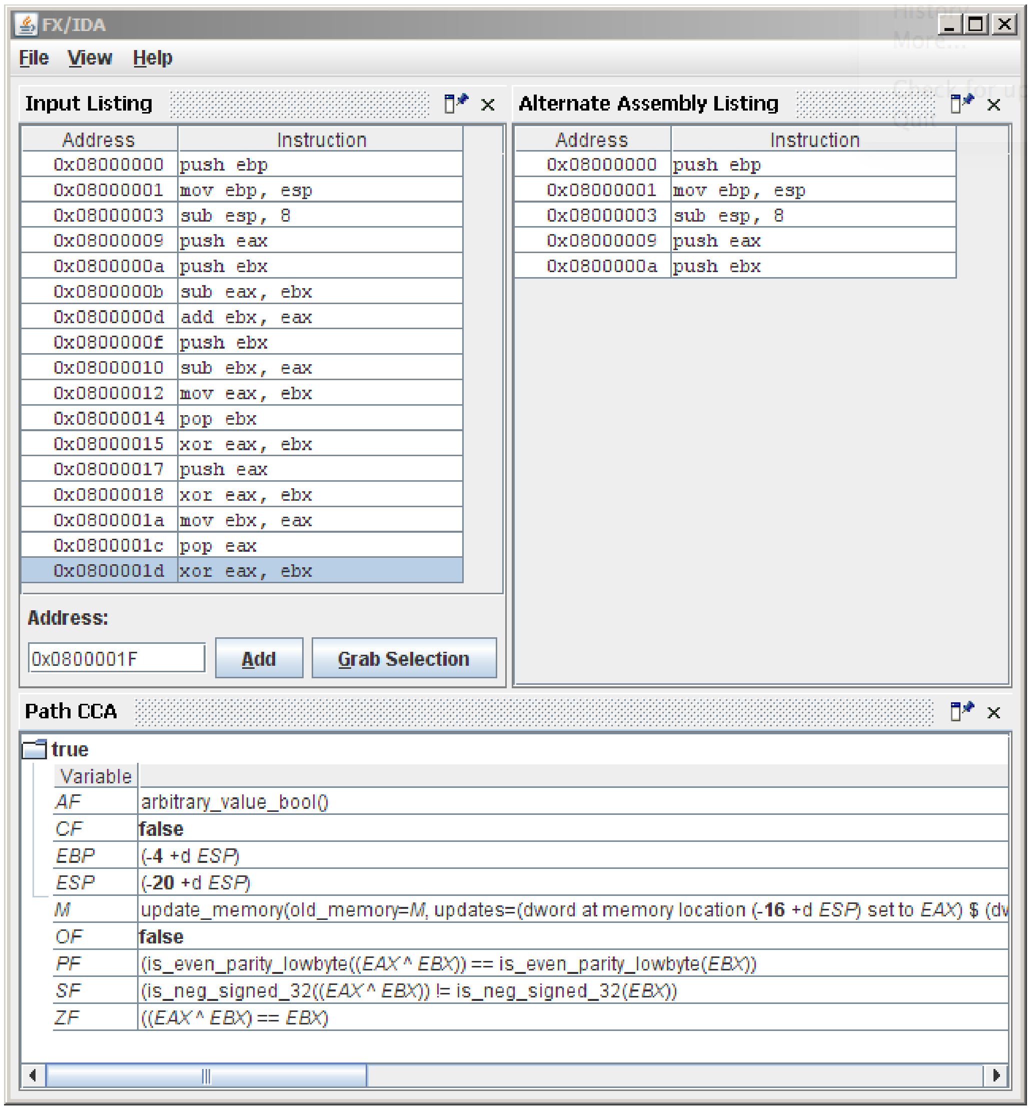
Address	Instruction
0x08000000	push ebp
0x08000001	mov ebp, esp
0x08000003	sub esp, 8
0x08000005	push eax
0x0800000a	push ebx
0x0800000b	sub eax, ebx
0x0800000d	add ebx, eax
0x0800000f	push ebx
0x08000010	sub ebx, eax
0x08000012	mov eax, ebx
0x08000014	pop ebx
0x08000015	xor eax, ebx
0x08000017	push eax
0x08000018	xor eax, ebx
0x0800001a	mov ebx, eax
0x0800001c	pop eax

Path CCA

true

Variable	
AF	arbitrary_value_bool()
CF	**false**
EAX	(EAX ^ EBX)
EBP	(-4 +d ESP)
ESP	(-20 +d ESP)
M	update_memory(old_memory=M, updates=(dword at memory location (-16 +d ESP) set to EAX) $ (d
OF	**false**
PF	(is_even_parity_lowbyte((EAX ^ EBX)) == is_even_parity_lowbyte(EAX))
SF	(is_neg_signed_32((EAX ^ EBX)) != is_neg_signed_32(EAX))
ZF	((EAX ^ EBX) == EAX)

Figure 2: Part 1 of a Miniature Example: FX/MC Behavior Computation Reveals no "No-Op" Code So Far



Input Listing

Address	Instruction
0x08000000	push ebp
0x08000001	mov ebp, esp
0x08000003	sub esp, 8
0x08000009	push eax
0x0800000a	push ebx
0x0800000b	sub eax, ebx
0x0800000d	add ebx, eax
0x0800000f	push ebx
0x08000010	sub ebx, eax
0x08000012	mov eax, ebx
0x08000014	pop ebx
0x08000015	xor eax, ebx
0x08000017	push eax
0x08000018	xor eax, ebx
0x0800001a	mov ebx, eax
0x0800001c	pop eax
0x0800001d	xor eax, ebx

Address: 0x0800001F

Alternate Assembly Listing

Address	Instruction
0x08000000	push ebp
0x08000001	mov ebp, esp
0x08000003	sub esp, 8
0x08000009	push eax
0x0800000a	push ebx

Path CCA

true

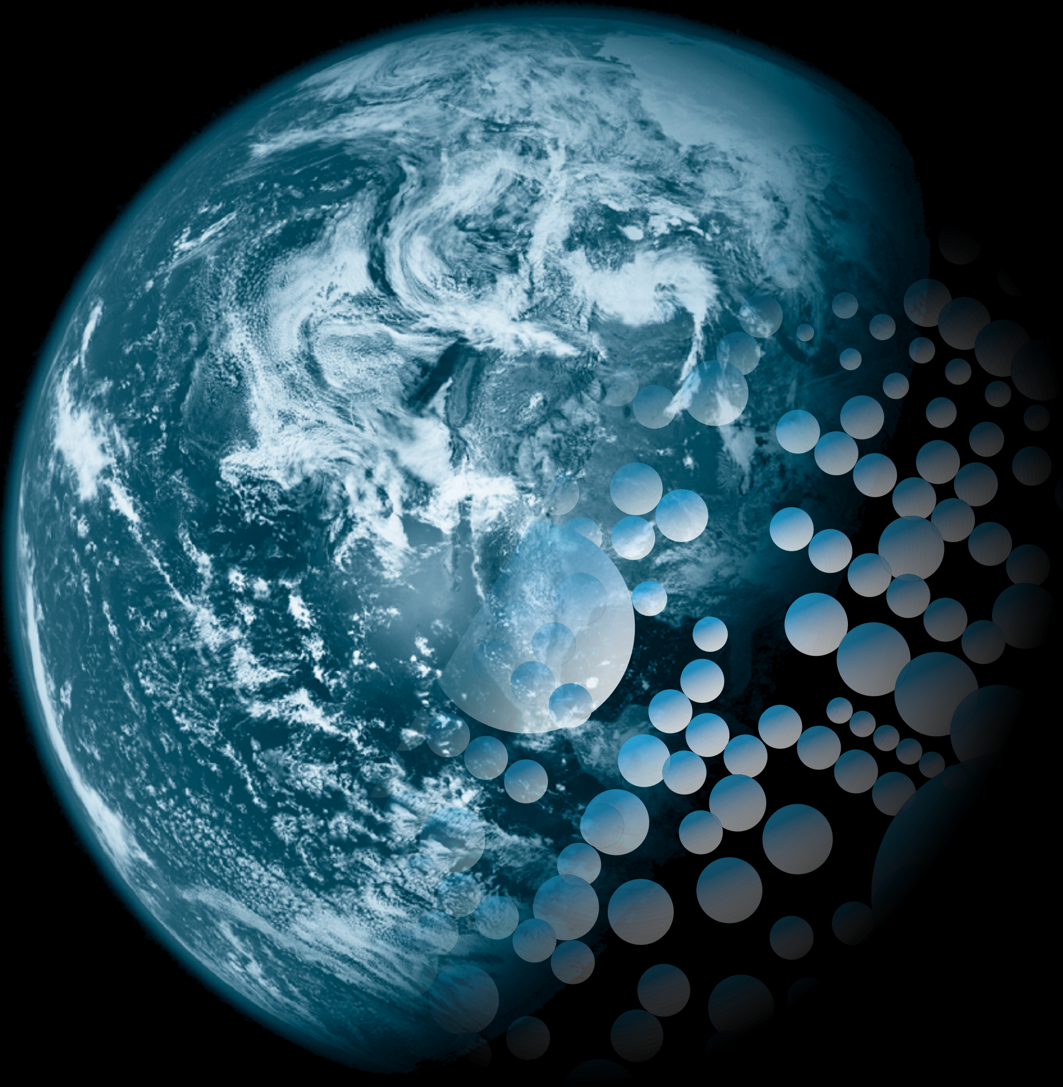
Variable	
AF	arbitrary_value_bool()
CF	**false**
EBP	(**-4** +d *ESP*)
ESP	(**-20** +d *ESP*)
M	update_memory(old_memory=*M*, updates=(dword at memory location (**-16** +d *ESP*) set to *EAX*) $ (dv
OF	**false**
PF	(is_even_parity_lowbyte((*EAX* ^ *EBX*)) == is_even_parity_lowbyte(*EBX*))
SF	(is_neg_signed_32((*EAX* ^ *EBX*)) != is_neg_signed_32(*EBX*))
ZF	((*EAX* ^ *EBX*) == *EBX*)

Figure 3: Part 2 of a Miniature Example: FX/MC Behavior Computation Detects and Eliminates No-Op Code

Cory F. Cohen

Jeffrey S. Havrilla

Function Hashing for Malicious Code Analysis



Function Hashing for Malicious Code Analysis

Problem Addressed

Program comparison continues to be an interesting field of research because it allows malicious code analysts to avoid duplication of effort and reduce the time spent analyzing malicious code. While much research has focused on the complex issue of roughly similar code [1], relatively little work has been done on quantifying the benefits of identifying nearly identical code. In our function hashing research, we laid a foundation for code comparison that dramatically reduces the size of the remaining code similarity problem.

Research Approach

Our approach was to use simple, high-confidence hashing algorithms to identify duplicated code in the CERT Artifact Catalog. The first technique was based on cryptographic hashing of program sections as identified in the Portable Executable (PE) header [2]. The second technique used a cryptographic hash of the bytes contained within a function, which we refer to as an *exact hash*. The third technique used a modified version of the function bytes to produce a *position independent code (PIC) hash.* These techniques proved very effective and identified with high confidence a large amount of code duplication within the malicious code sample set.

When we began our research, there were approximately 4 million executable files in the CERT Artifact Catalog. Each file was processed to parse the PE header and identify the bytes contained in each program section. These sections typically correspond to the code, data, imports, and other portions of a Windows executable file. An MD5 cryptographic hash was computed for the bytes of each section, and the results were analyzed to determine which sections were shared among different executable files in the catalog.

In addition to clustering similar programs that shared common sections, this analysis identified the most common cause of whole-file MD5 hash differences between code-identical programs. Approximately half of the samples were found to be duplicates of other files when the “slack space” was excluded when calculating MD5 hashes. Slack space refers to the bytes at the end of an executable file that are not referenced in the PE header and are not loaded into memory by the program loader. Malicious programs frequently use this space for configuration or other data by reading the bytes into memory during execution. Code-identical programs excluding the slack space are very likely to be identical malware with different run-time data.

By excluding the hashes for the slack space and PE header, we were able to reduce the number of unique programs in the sample set by almost 50 percent. The remaining two million files were processed by a loosely-coupled cluster of approximately 40 virtual machines running IDA Pro to identify the function byte boundaries. Slightly more than 757 million function instances were found, averaging 394 functions per file. IDA Pro’s detection of function boundaries is not perfect, and it is possible that the derivative results are biased by this incorrect identification, although this bias is not believed to be significant.

We produced an exact hash of the bytes between the beginning and end of each function using MD5. Only 154 million unique exact hashes were found in the data set, representing a reduction of nearly 5 to 1 over the number of function instances identified. Functions sharing the same exact hash are almost certainly the same function, and because memory references are included in the hash, it is very likely that they are in fact the same function at the same address in a nearly identical program. This level of exact hash duplication suggests that our malware sample set has many nearly identical programs. This is consistent with the significant reduction encountered in section hashing.

Identical functions are frequently found at different locations in memory since the compiler is usually used for generating the memory layout of a program from the source code. To normalize for frequent relocation of functions, we developed a position independent code (PIC) hash by replacing some of the bytes supplied to the MD5 algorithm with zeros. The zeroed bytes in the PIC hash were identified as follows:

- If IDA identified an operand for an instruction as a memory reference, then the byte corresponding with that operand was replaced with zeros.
- Any immediate operands that were valid memory addresses within the program were also zeroed.
- Instruction op-codes, relative jumps, register operands, and other immediate values were left unchanged.

We expected that this PIC algorithm would produce multiple exact hashes for a given PIC hash, and a single PIC hash for a given exact hash. In practice one design flaw and one implementation defect caused about 1.27 percent of the PIC hashes to map to multiple exact hashes. The design flaw involved truncated programs with immediate operands that referenced the truncated part of the executable. In these cases we failed to zero the immediate operand bytes because IDA Pro correctly reported that the operand was not a valid address in the truncated file. The implementation defect caused data bytes within the function boundaries to be excluded from the PIC hash, but not from the exact hash. We were able to make approximate corrections for these bugs in the data set without regenerating all of the function hashes.

We identified over 39 million unique PIC hashes, reducing the function space by nearly 4 to 1 compared to the unique exact hashes, and nearly 20 to 1 compared to the function instances. Figure 1 shows the reduction in the number of function instances to be analyzed, starting with approximately 1.5 billion functions in the original dataset, dropping to 757 million functions after the section hash techniques were used to remove nearly duplicate programs, and further declining to 154 million and then 39 million as the two function hashing techniques were applied.

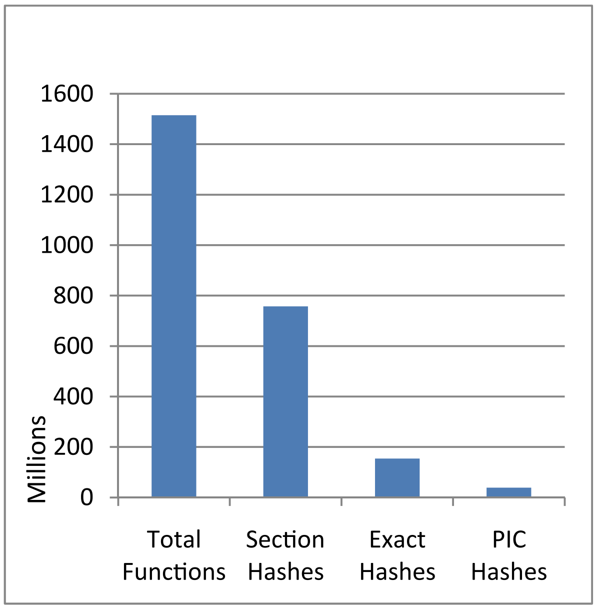


Figure 1: Function Instance Reduction

While functions sharing a given PIC hash are less likely to really be the same function than those sharing the same exact hash, in practice we have found that all functions with a given PIC hash are the same function, only at a different address in memory. Some proportion of PIC hashes are known to have different interpretations depending on the context in which they occur. For example, we found two PIC hashes that correspond with nearly 83 million function instances. These two PIC hashes are produced by two variants of the Intel jump instruction, which are commonly referred to as "thunks" (functions which simply jump to another function). It is likely that other semantically different functions share the same PIC hash, especially when the code is small and relatively mundane. We have found object constructors and exception handling code with multiple context dependent variants that produce the same PIC hash. Despite these contextual issues, most PIC hashes appear to correlate with unique function5 behavior.

Additional Analysis

The exact and PIC function hash data has proven to be a rich source of data for additional derivative research. In addition to preventing duplicative analysis of malicious functions, identifying known library functions and other uninteresting code is even more beneficial. Just providing analytically meaningful names for as many function hashes as possible can have real operational impact. We collected names assigned by the IDA FLIRT system [3] and correlated those results with the exact PIC function hashes in our data set. The FLIRT system was found to be more reliable than we anticipated, with less than 1 percent of exact hashes and less than 4 percent of PIC hashes with substantially inconsistent names. This step alone labeled approximately one-third of the function hashes. While IDA may have assigned a consistent name to given hashes, or "a given hash" not all occurrences of the hash may have been named across different files. By assigning names to all occurrences of a given hash, we were able to improve on IDA's FLIRT system by over 60 percent.

We also conducted some experiments attempting to identify "high-value" functions that were of interest to malicious code analysts. In this way, we were able to identify nearly 5,000 PIC hashes and over 14,000 exact hashes that involved cryptographic hash routines used in malicious code. We then correlated these hashes with the programs that used them, and produced trend data showing the relative growth of specific hash routines in malware over the last several years (see Figure 2).

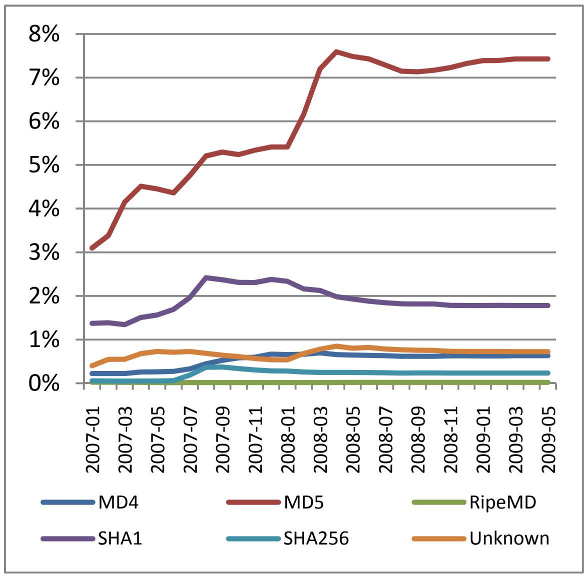


Figure 2: Cryptographic Hash Prevalence in the CERT Artifact Catalog

The function hash data is also very useful for clustering malware samples into families. We divided the functions into two categories: those that occurred only in specific families (and thus were indicative of that family) and those that occurred across multiple families (and are more likely to be library functions or other uninteresting code). Once the functions were divided in this way, clustering was a simple matter of grouping all programs that shared indicative functions, which is easily accomplished in linear time for even large data sets of several million files.

Since this approach required prior knowledge of the families to divide the functions, we used the Pithos tool described in last year's research report [4] to bootstrap the process. This resulted in a very small number of PIC hashes (about 7 percent) being marked as non-indicative. These hashes represented more than 91 percent of all function instances, strongly supporting the assertion that these functions were primarily library functions or other highly duplicative code.

Benefits

We have used simple hashing techniques to dramatically reduce the size of the malicious code similarity challenge and provide other valuable insights into malicious code behavior. While other more advanced techniques may be required to gain a complete understanding of malicious code similarity, new problems faced by malicious code researchers will be more tractable in the future.

2009 Accomplishments

In 2009, we made significant progress in reducing the size and complexity of the malicious code similarity challenge by reducing the number of functions requiring analysis by a factor of 40. Simple hashing algorithms were shown to provide value by identifying large numbers of reused and duplicated functions, preventing the unnecessary duplication of analyst effort when reverse engineering malicious code. Function hashing techniques were able to provide new methods of clustering malware families. Our understanding of the contents of the CERT Artifact Catalog has improved, leading to additional insight into the malicious code threat landscape, and many new avenues of research into malicious code clustering and program comparison have been opened to further research.

2010 Plans

We plan to characterize the nature of the function hashes by assigning names to them. Our initial results are summarized in Figure 3, which shows the approximate proportions of different types of functions. Each pie chart shows the functions named by IDA FLIRT, the functions indicative of a specific malware family, the functions which might be named using import profiling techniques still under development, and finally those functions that remain uncategorized. While large percentages of the function instances remained unnamed, they represent very small percentages of the unique hashes. This suggests that those functions are heavily re-used, and might be easily named using techniques similar to IDA's FLIRT capability.

We plan to extend the existing research to include more sophisticated identification techniques based on flow control similarity, heuristic similarity, and semantic equivalence. This will further reduce the size of the code comparison problem, and enable more complex views of the Artifact Catalog of more interest to malicious code analysts.

References

[1] Willcock, C., Saebjornsen A., Panas, T., Quinlan, D., Su, Z., "*Detecting Code Clones in Binary Executables*" http://portal.acm.org/citation.cfm?id=1572272.1572287

[2] Microsoft Corporation, "*Microsoft Portable Executable and Common Object File Format Specification*", http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

[3] Guilfanov, I., "Fast Library Identification and Recognition Technology", http://www.hex-rays.com/idapro/flirt.htm

[4] Cohen, C., Havrilla, J., "*Malware Clustering Based on Entry Points*", CERT Research Annual Report 2008 http://portal.acm.org/citation.cfm?id=1572272.1572287

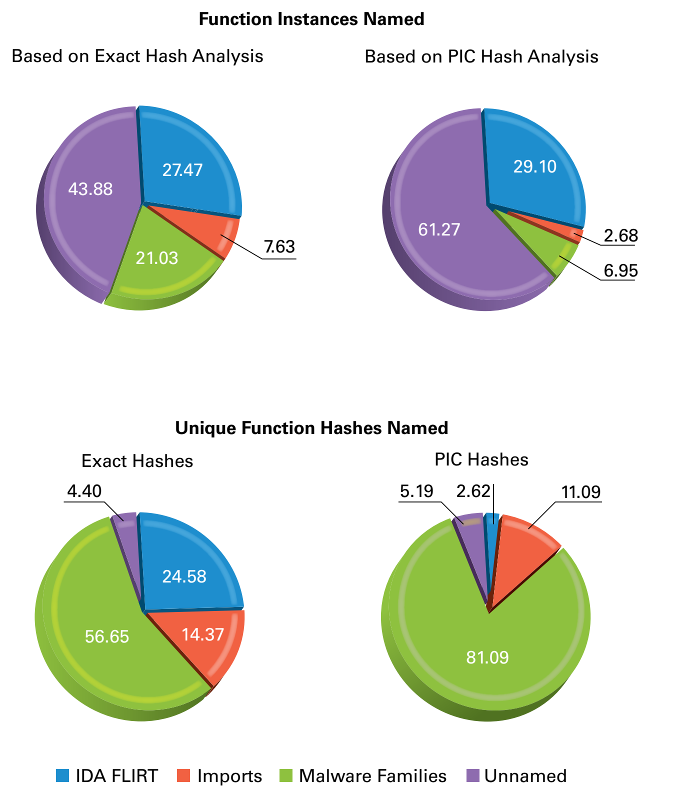
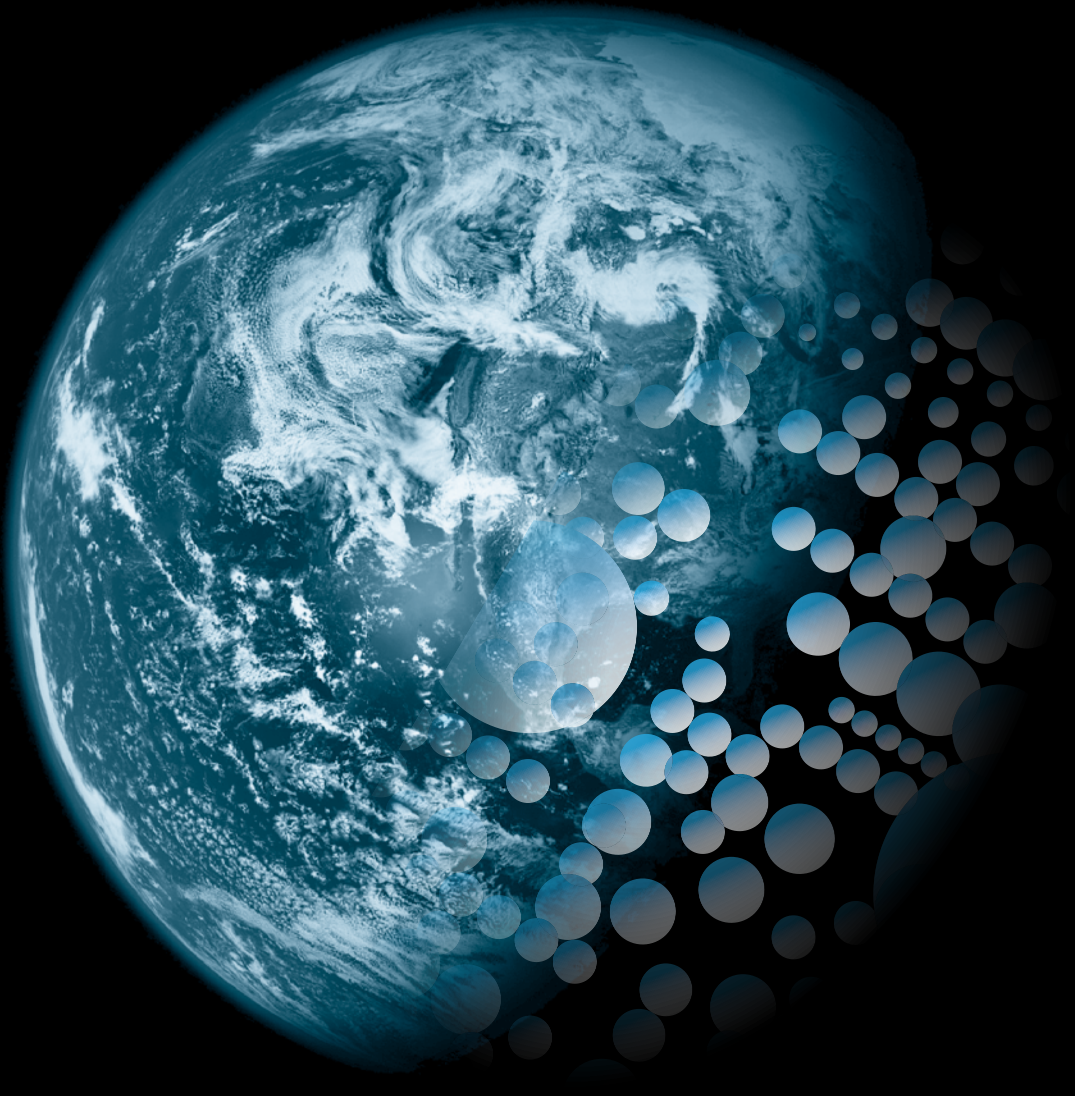


Figure 3: Preliminary Results for Categorizing Functions

Soumyo D. Moitra

Evan Wright

Metrics for Evaluating Network Sensor Placement



Metrics for Evaluating Network Sensor Placement

Network sensors collect, detect, or help mitigate malicious activity on network traffic data. Metrics for evaluating their benefits at a given location can be improved. The improvements in the metric help in prioritizing sensor deployment and improve network security for an organization, given a budget constraint.

An existing metric baseline was improved by two implementation approaches. One approach consisted of examining the metric along with its weights, and then suggesting improvements. Based on expert opinion, suggestions were made to modify the existing metric to improve its accuracy. The other approach consisted of identifying new measures that could be included to form a more comprehensive metric. Four new measures were constructed to augment the metric using network data. The result is a new metric that should better reflect the benefits from deploying a sensor at any particular location.

Problem Addressed

A major challenge for information security decision makers is to develop better metrics that help evaluate the benefits of deploying network sensors. Sensors are critical to providing security in networks. They help by detecting attacks, intrusions, and malicious software on computer networks. Sensors can be of various types, which include anomaly-based intrusion detection systems, signature-based intrusion detection systems, and intrusion prevention systems. These sensors can alert security personnel of potential threats and may also help mitigate threats. It is generally not possible to purchase all the sensors an organization may desire because of resource limitations. Therefore, a prioritization scheme has to be employed to decide the number of sensors and the locations to deploy them so that, given the constraint on resources, the security provided by the sensors can be maximized. These metrics for the prioritization scheme should accurately represent the advantages gained by purchasing sensors for specific locations.

An effective metric should be as comprehensive and objective as possible. For large organizations with many sub-organizations, it is difficult for decision makers to completely understand the operating environment of all the sub-organizations. The sub-organizations may have a deeper insight into their own situation and needs, but the decision makers may be biased and also may be unfamiliar with the budgetary constraints of the organization as a whole. Often, metrics exist that have not been systematically analyzed. In such cases, there is a need to review these metrics and then develop a new and improved metric. This new metric can be used by the overall organization to prioritize the needs of the sub-organizations and make the sensor deployments more effective.

Research Approach

We use the following terminology:

- **Measures:** components of the sensor metric that are related to the criticality of the network of the sub-organization that wishes to deploy a sensor to cover its network
- **Scores:** the numeric output from the new measures constructed here - diversity, visibility, etc.
- **Values:** magnitudes of the importance weights associated with the measures

The existing sensor evaluation metric was primarily focused on the evaluation of sub-organizational criticality.

The measures used in this metric were relevant to the benefits of placing a sensor at a particular location. Specific sub-organizations that house mission-critical data or that play a mission-critical role were prioritized in this metric. The secondary focus of the metric was to quantitatively identify the security policy exceptions present in each sub-organization. Each measure had an importance weight associated with it. The measures and their importance weights were combined to form the composite metric. Therefore this composite metric represented the overall benefit of deploying a sensor at that location.

Two sources of information, which are not often utilized, can help an organization optimize this metric. The first is the set of experts across organizations that have technical experience related to the specific threats that they operationally face every day. The second is a set of network-based tools that are used to monitor network traffic. In this research, both these information sources were used to develop an improved metric.

The first part of this work focused on

- studying the existing metric for relevance and completeness in terms of the existing measures used
- benchmarking with comparable models that have been established in the area of decision analysis; in particular, the scales were investigated to verify that the proper measures and importance weights were used
- investigating issues regarding the correlations among measures used in the metric
- identifying the modifications and possible enhancements that might improve the metric
- verifying the actual weights used: this entailed designing a survey to elicit expert opinion and analyzing the data from the survey
- suggesting several changes to the importance weights

In the second part, some new measures based on network traffic data were constructed to augment the existing metric. This was done by

- conducting empirical analyses of traffic based on network flow data; network flow is used by telecommunications service providers and other large organizations for large-scale network traffic analysis
- developing tools and methodologies to capture and summarize the network flow data
- computing flow volumes in terms of connections, packets, bytes, timing, and other information
- constructing a set of measures based on this network flow data that indicate future effectiveness of sensors by location

For each sub-organization, a specific sampling of data was collected and evaluated by an average of four measures

1. Total volume in packets: a ratio of the sub-organization's packet volume to the whole organization's packet volume.
2. Diversity: a value that captures the number of service requests of the sub-organization, so that its relative value can be compared to other organizations' scores.
3. Visibility: the percentage of IP addresses on the Internet that communicates with the sub-organization.
4. Ratio of encrypted to unencrypted traffic: network sensors cannot inspect encrypted payloads, and little added value is obtained from sensors when most packets are encrypted.

Expected Benefits

The new metric we have developed includes inputs from experts and data-derived measures that will lead to better decisions regarding sensor deployment. This should provide better security for the budget of a given organization. The inputs from experts have helped to verify, and in some cases modify, the importance weights for the existing measures that go into estimating the overall metric. These inputs reflect a consensus among experts who have been involved with network sensors, and this provides a better indication of the benefits of having a sensor at a given location.

By using network traffic data, we will be able to avoid the problems in decision making based on monetary values. The scores derived from flow data are objective measurements, where no disagreement exists about their values given their definitions. This empirical approach avoids vertical organizational contention where decision makers at the top may not be fully aware of operational impacts on the sub-organizations. Moreover, it would reduce any perception of bias in middle management's judgment concerning any subordinate organization.

2009 Accomplishments

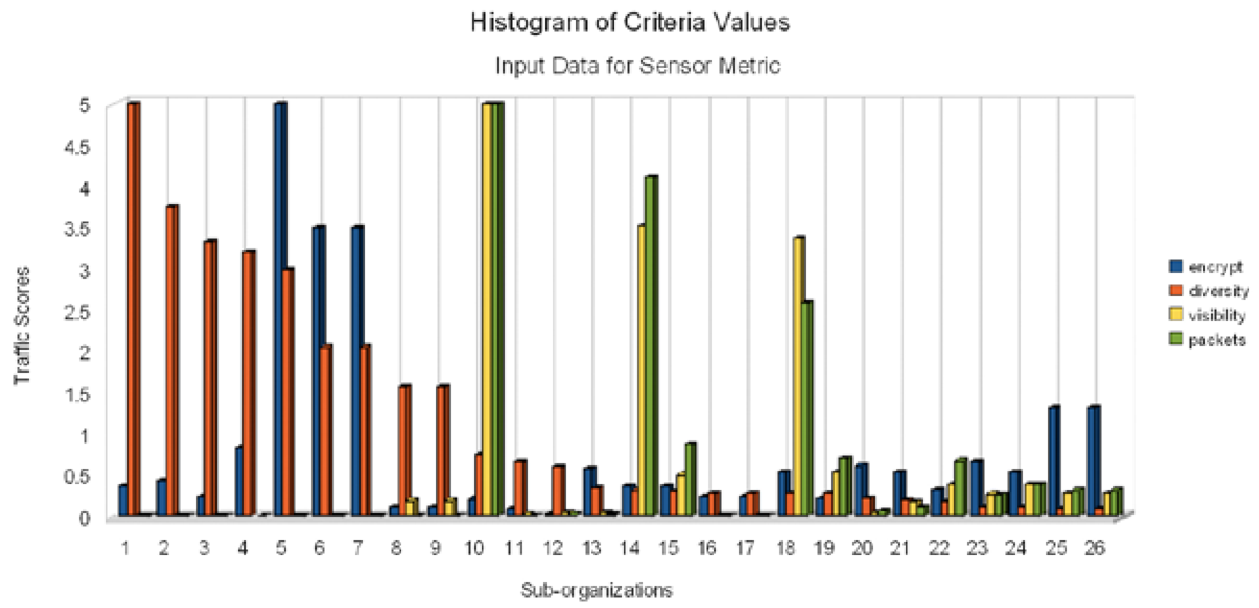
This project demonstrates how the use of experts can be used to improve importance weights and how their values can be verified and modified based on expert opinion. This was done through a survey and then analyzing the data from that survey.

For the analysis using network data, we have designed software that will automatically evaluate each of the four measures. We have added a stipulation to the sensor application process that requires a list of IP addresses from the sub-organizations that desire funding for security purchases. Next, we have successfully automated all the data collection, manipulation, organization, and statistics generation of network flow data. The numeric output consists of prioritization values that measure the benefits of placing a sensor inside each sub-organization.

2010 Plans

We will be able to improve future metrics by gathering and analyzing results from the 2009 sensor metric analysis. Our new testing data will be more precise than previous testing data, allowing for a more objective metric in the next revision. This will also allow a more evenly designed partitioning system that will better distribute criteria scores across the scales that are used by the metric.

Our domain knowledge of the network flow data can be applied to reduce noise in the network analysis before each of the four criteria is calculated. Noisy traffic may include irrelevant scans that do not affect security. This will allow us to have a more accurate measurement of the network traffic that would be applicable to decision making for purchasing sensors.

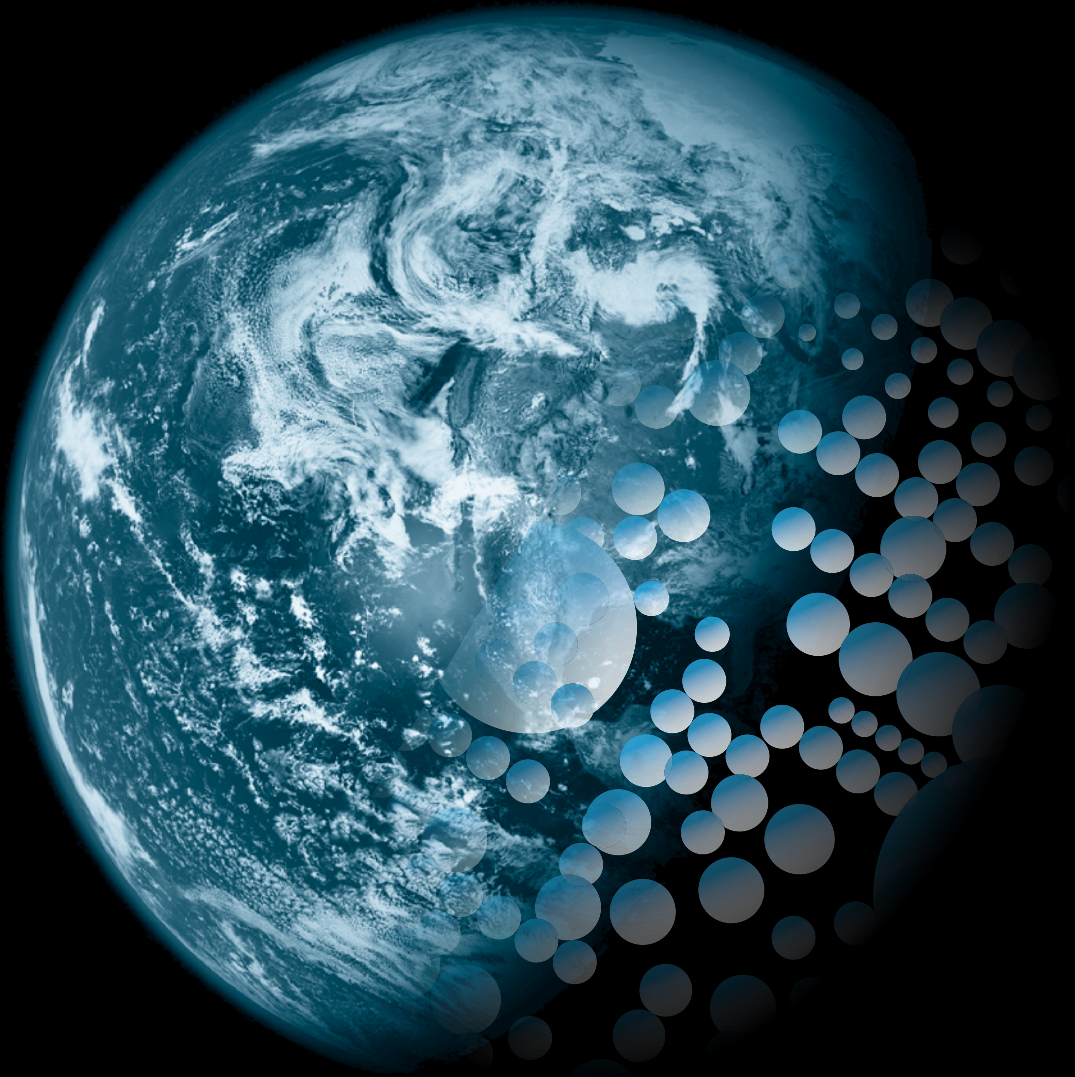


Andrew Moore

Dawn Cappelli

Randy Trzeciak

Modeling Insider Theft of Intellectual Property



Modeling Insider Theft of Intellectual Property

Problem Addressed

Insider theft of intellectual property (IP) is a real concern in today's organizational environment. These insiders often plan to develop a competing product or use the stolen information to lure clients from the victim organization. They may conspire with a competitor as a prelude to employment with that organization, or simply sell IP to a competitor.

Cases documented in the CERT Insider Threat Database illustrate the problem. For instance, the head of the public finance department of a securities firm recruited employees to collect documents to take to a competitor. Then he sent a resignation letter to the head of the sales department on behalf of himself and each of the employees he recruited to his scheme. The following week, the entire group started work with the competitor. In another case, an outsider operating a fictitious company recruited a job-seeking employee to send him reams of his current employer's IP by email, postal mail, and commercial carrier.

Since 2001, CERT has researched malicious insider incidents, including IT sabotage, fraud, theft of confidential or proprietary information, espionage, and threats to U.S. critical infrastructure. The consequences of these incidents include financial loss, operational impact, damage to reputation, and harm to individuals. In fact, the actions of a single insider have caused damage ranging from lost staff hours to public relations and financial damage so extensive that businesses have been forced to lay off employees or cease operation. Furthermore, insider incidents can have repercussions beyond the affected organization, disrupting operations or services critical to a specific sector or creating serious risks to public safety and national security.

CERT's Management and Education of the Risk of Insider Threat (MERIT) work uses empirical data CERT collects to outline the complexity of insider events—especially the unintended consequences of organizational culture, policies, practices, technology, and efforts to manage insider risk. MERIT employs system dynamics modeling and simulation to better grasp and communicate the threat to information technology (IT) systems posed by malicious current or former employees and contractors. Our past work has involved modeling insider fraud [1], insider IT sabotage [2], and national security espionage [3].

Based on our initial modeling work and case analyses, we have found that different classes of insider crimes exhibit different patterns of problematic behavior and mitigative measures. CERT identified four categories of insider threat based on patterns observed in the cases analyzed: IT sabotage, theft or modification of information for financial gain (fraud), theft of IP, and national security espionage. We believe modeling these crimes separately can be more illuminating than modeling the problem as a whole. Our modeling effort in 2009 focused on understanding, documenting, and analyzing the behavioral and technical patterns of crimes involving insider theft of IP [4].

Research Approach

Our research approach is based on the comparative case study methodology [5]. The cases we selected fit our definition of insider theft of IP.[1] We identified these cases through public reporting and included primary source materials, such as court records in criminal justice databases (found through searches on Lexis court databases) and secondary source materials, such as media reports (found through searches on Lexis-Nexis news databases and Internet search engines such as Google). We used the following criteria to select cases:

- The crime occurred in the United States.
- The subject of the crime was prosecuted in a United States court (possibly resulting in a plea bargain).
- The available data were of sufficient quantity and quality to understand the case.

We identified and analyzed 48 cases of IP theft meeting these criteria. While generalizing from case study comparisons in general, and from our study in particular, is problematic, this method does improve understanding of the context surrounding and influencing the event. Indeed, the sole purpose of our modeling effort is to help people understand the complex nature of the threat. Our models evolved through group data analysis sessions involving experts in both the behavioral and technical aspects of insider crimes. This analysis used system dynamics, a method for modeling the holistic behavior of complex problems as they evolve. System dynamics provides insight to difficult management situations in which efforts to solve a problem actually make it worse.

System dynamics model boundaries are drawn so that all the variables necessary to generate and understand problematic behavior are contained within them. This approach encourages the inclusion of soft factors, such as policy-related, procedural, administrative, or cultural factors. Arrows represent the pairwise influence of the source variable on the target variable. A solid arrow indicates that the values of the variables move in the same direction, whereas a dashed arrow indicates that they move in the opposite direction.

In system dynamics, the dynamic complexity of problematic behavior is captured by the underlying feedback structure of that behavior. Our models identify two types of feedback loops: balancing and reinforcing. Balancing loops (indicated by the letter B followed by a number) drive variables to some goal state and are typified by aspects that control problematic behaviors.

1 We define insider theft of IP as crimes in which current or former employees, contractors, or business partners intentionally exceeded or misused an authorized level of access to networks, systems, or data to steal confidential or proprietary information from the organization.

Reinforcing loops (indicated by the letter R followed by a number) describe system aspects that drive variable values consistently upward or downward and are typified by escalating problematic behaviors.

Expected Benefits

We believe our models will enable a better understanding of the complex nature of this threat. Improved understanding stimulates better awareness and intuition about the effectiveness of countermeasures. For instance, our models reveal behavioral and technical precursor activities that might alert an organization to the potential for insider theft of IP. Understanding the relevance of these precursors provides organizations an opportunity to detect IP theft prior to employee termination. The model suggests monitoring for the following activities:

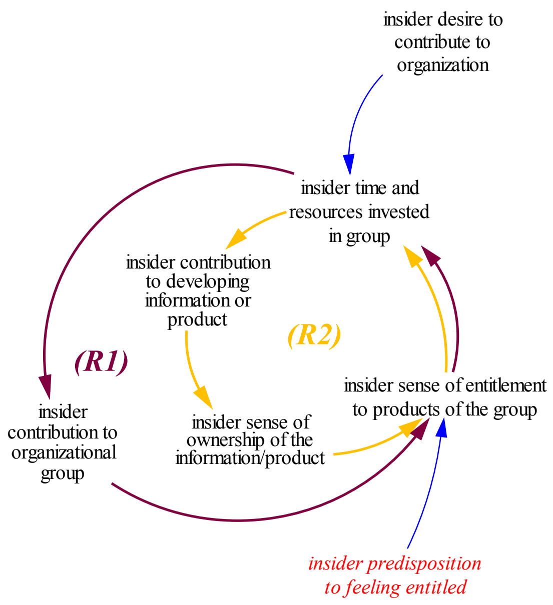
- deceptive practices
- attempts to use IP (e.g., to sell the IP or solicit business from customers of the victim organization)
- attempts to gain increased access to sensitive information
- downloading of information in excess of normal patterns
- emailing of attachments outside of normal patterns
- monitoring of backup tape use
- exfiltration of information within 30 days of resignation

2009 Accomplishments

Our work identified two dominant models in the insider theft of IP cases: the *Entitled Independent Scenario* and the *Ambitious Leader Scenario*. Entitled Independents are ambitious insiders acting alone to steal information to take to a new job or side business. They feel a sense of entitlement to that IP based on their participation in its development. This sense of entitlement can be reinforced by an event they perceive as dissatisfying. Insiders then used stolen IP as leverage to pursue new opportunities. The Entitled Independent is more often than not fully authorized to access this information and steals it when nearing resignation. Sometimes, he or she will engage in deception to facilitate the theft. Entitled Independents rarely act as if they are doing anything wrong. Figure 2 presents the part of the Entitled Independent model concerning the escalation of entitlement.

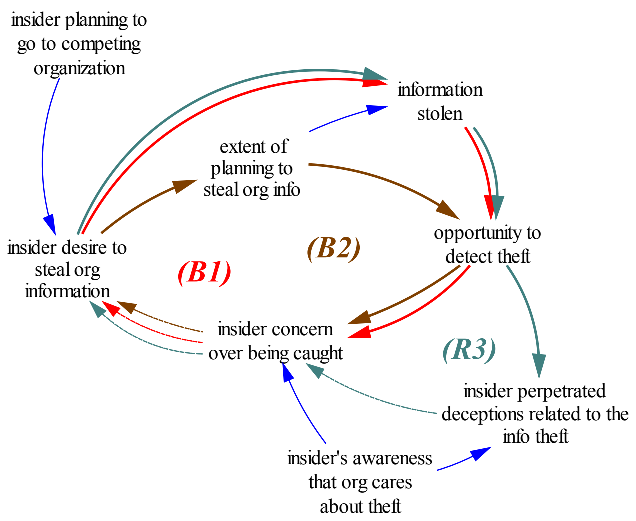
In the Ambitious Leader scenario, the insider recruits other insiders to steal information for some larger purpose. The cases can be distinguished according to whether the insider

- had plans to develop a competing product or use the IP to lure clients from the victim organization
- was working with a competing organization to help his new employer
- sold the information to a competing organization



insider desire to contribute to organization
insider time and resources invested in group
insider contribution to developing information or product
(R2)
(R1)
insider sense of entitlement to products of the group
insider contribution to organizational group
insider sense of ownership of the information/product
insider predisposition to feeling entitled

Figure 1: Insider Entitlement Escalation



insider planning to go to competing organization
information stolen
extent of planning to steal org info
opportunity to detect theft
insider desire to steal org information
(B2)
(B1)
insider concern over being caught
(R3)
insider perpetrated deceptions related to the info theft
insider's awareness that org cares about theft

Figure 2: Theft Planning by Ambitious Leader

This scenario also describes cases in which the insider was partially motivated by a desire to help a foreign government or company. It is more complex than the Entitled Independent scenario, involving more intricate planning, deceptive attempts to gain increased access, and recruitment of other employees to the leader's scheme. Figure 3 presents the part of the Ambitious Leader model concerned with theft planning.

In addition to the two system dynamics models for insider theft of IP, we developed a workshop on insider threat that includes interactive exercises. These exercises help participants assess their own organization's vulnerability to insider threat in specific areas. We want participants to leave the workshop with actionable steps they can take to better manage the risk of insider threat. Because the sensitivity of actual case data precludes its use for training, we created a representative fictional case around the Ambitious Leader scenario. This fictional case illustrates and communicates the key aspects of insider crimes.

2010 Plans

Future work will refine models of insider fraud and national security espionage. Using system dynamics, we will attempt to assess the weight and interrelatedness of personal, organizational, social, and technical factors. Of course, our ultimate concern is to develop effective countermeasures to insider crimes, and we plan to use modeling and simulation to identify and evaluate the effectiveness of such measures in the workplace.

We recognize methodological and data challenges must be overcome before research on insider activity can effectively prescribe mitigation policies, practices, and technology. Prospective studies of these phenomena will always be challenging because of low base rates. However, we believe system dynamics modeling using available empirical data can bridge this methodological gap.

References

[1] E. Rich, I.J. Martinez-Moyano, S. Conrad, D.M. Cappelli, A.P. Moore, T.J. Shimeall, D.F. Andersen, J.J. Gonzalez, R.J. Ellison, H.F. Lipson, D.A. Mundie, J.M. Sarriegui, A. Sawicka, T.R. Stewart, J.M. Torres, E.A. Weaver, J. Wiik, "Simulating Insider Cyber-Threat Risks: A Model-Based Case and a Case-Based Model," in Proceedings of the 23rd International Conference of the System Dynamics Society, July 2005. http://www.cert.org/insider_threat/docs/insider_threatISDC2005.pdf

[2] Moore, A. P., Cappelli, D. M., & Trzeciak, R. F. (2008). The „Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures (Vol. Insider Attack and Cyber Secruity: Beyond the Hacker). (S. Stolfo, S. M. Bellovin, S. Hershkop, A. Keromytis, S. Sinclair, & S. W. Smith, Eds.) New York, NY: Springer Science+Business Media, LLC.

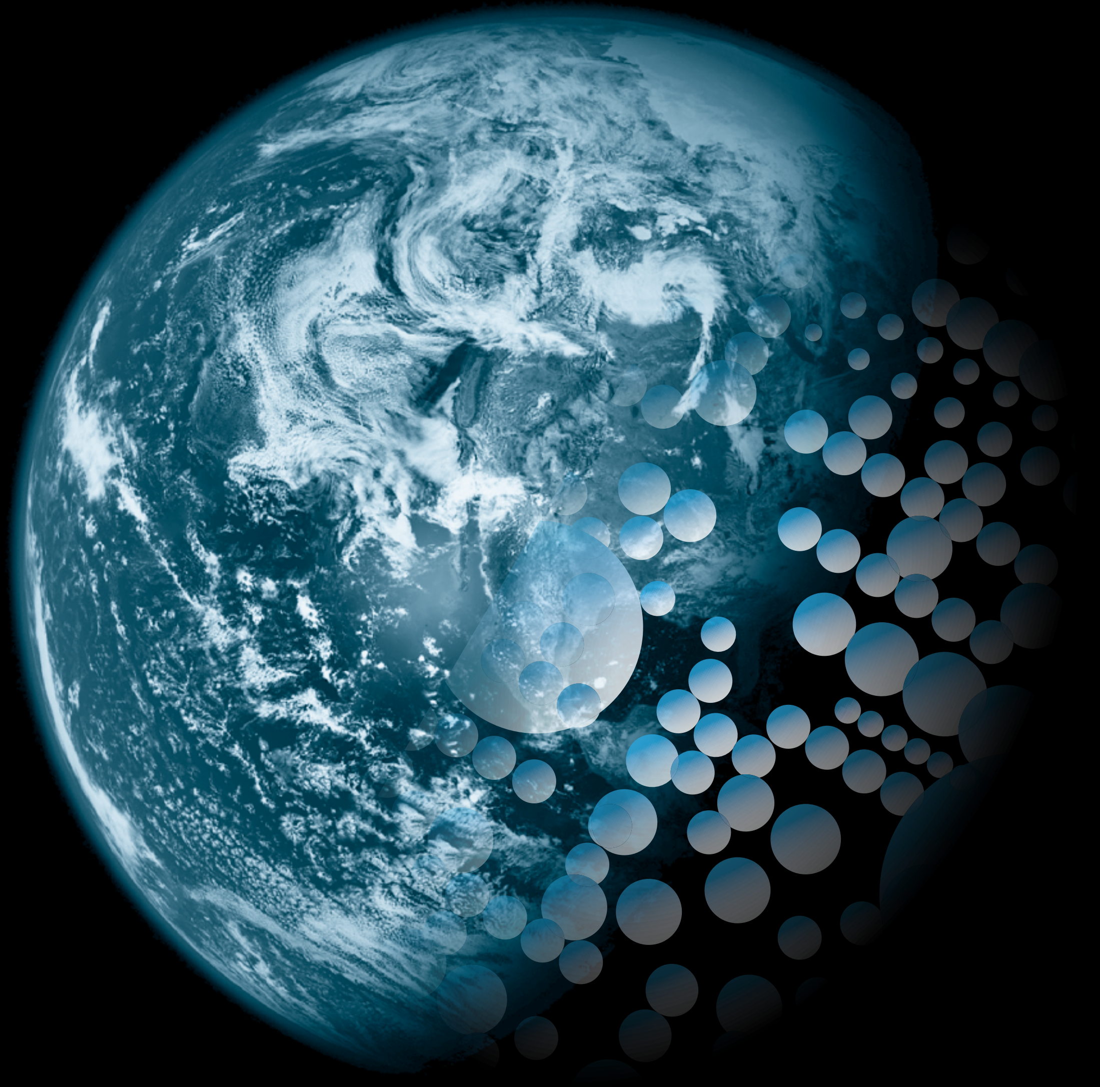
[3] Band, S. R., Cappelli, D. M., Fischer, L. F., Moore, A. P., Shaw, E. D., & Trzeciak, R. F. (December 2006). Comparing Insider IT Sabotage and Espionage: A Model-Based Analysis. Carnegie Mellon University, Software Engineering Institute.

[4] Moore, A.P., D.M. Cappelli, T. Caron, E. Shaw, R.F. Trzeciak, "Insider Theft of Intellectual Property for Business Advantage: A Preliminary Model," in Proc. Of the 1st International Workshop on Managing Insider Security Threats (MIST2009), Purdue University, West Lafayette, USA, June 16, 2009. http://www.cert.org/insider_threat/docs/Insider_Theft_of_IP_Model_MIST09.pdf

[5] Yin, R. K. (2003). *Case Study Research.* (3rd, Ed.) Thousand Oaks: Sage Publications.

Rayon: A Unified Framework for Data Visualization

Phillip Groce



Rayon: A Unified Framework for Data Visualization

Introduction

Data visualization summarizes large volumes of data and represents this data pictorially. Data visualization is used in a wide variety of applications, but visualization techniques that are effective in one application can often be used as well or better in another application. When organizations depend on good data visualization, a unified visualization capability will often increase that effectiveness; this is especially important if an organization relies on internal experts to create new visualization techniques appropriate to their environment. The Rayon visualization toolkit was developed to augment large-scale network analytic information, and to improve the visualization capability and productivity of analytic operations by making it possible to share visualization techniques between applications.

Problem Addressed

To be effective, network security analysts must quickly analyze large volumes of network monitoring data. Network analysts often specify custom visualizations in order to meet this challenge. These visualizations are applied to data to produce graphics, which highlight aspects of network activity the analysts regard as important.

Graphics may be produced for several purposes, for example

- Exploration: An analyst generates a graphic to better understand a specific problem. The graphic is part of an iterative process of understanding the data. An interactive graphic may be useful to iteratively "walk through" the dataset.
- Explanation: An analyst generates a graphic to illustrate a point. The data for the graphic is well understood and carefully chosen. The graphic may be heavily annotated. The graphic may be printed, or included in a slide presentation.
- Validation: A scheduled process generates a graphic periodically as part of ongoing reporting and situational awareness. Analysts have expectations about the data, and the graphic either validates or highlights violations of these expectations. When expectations change, a series of validating graphics may document a transition from one set of expectations to another.

These use cases have diverse requirements concerning (among others) how quickly the graphic can be generated; whether the output can be saved or shared; and how much user attention is required to generate them. However, core visualization techniques such as scatterplots, bar charts and line plots can often be shared within these use cases. When an organization develops a new visualization technique or useful combination of existing techniques, it can be most effectively used when it is available for all these use cases.

Several software packages provide visualization capabilities. Examples are R [1], Matlab [2] and Microsoft Excel [3]. Sometimes visualization is provided as part of a larger analytic tool. Other tools exist only to do visualization. All of these packages simplify the process of data visualization. However, different packages have different strengths, and using multiple packages presents problems:

- No common language exists that applies to all visualization tools. Visualization techniques developed in one tool cannot be migrated to another without additional software effort. Similarly, different tools may export graphics in different ways. Some tools may support highly interactive graphics, but little to no export functionality.
- Visualization tools often depend on additional software. These dependencies differ from package to package, and may not be installed or available. The additional packages may complicate configuration management and lengthen time to deployment (e.g., by requiring additional change control requests).
- Default options are inconsistent between tools.

Ideally, an organization's visualization capabilities would be integrated to provide a consistent user interface and portable visualization techniques.

Research Approach

The Rayon visualization toolkit has been developed to provide a single platform that can be used to generate graphics in data exploration, explanation, and validation applications. Rayon provides an application programming interface (API) in the Python [4] programming language for specifying visualizations, and a facility for generating graphics from these specifications using different low-level software packages. Rayon can currently render static graphics using the Cairo [5] graphics library or interactive graphics using the WxWidgets [6] GUI toolkit.

In Rayon, a visualization developer (e.g., a network analyst who wishes to view a type of data in a particular way) specifies a visualization by creating an object called a *Plot*. Rayon comes with several common types of Plots. Plots can be overlaid onto each other or tiled; visualization developers may therefore compose more complicated Plots from numerous simpler Plots. For example, a line plot of trended data may be laid over a scatterplot of raw data.

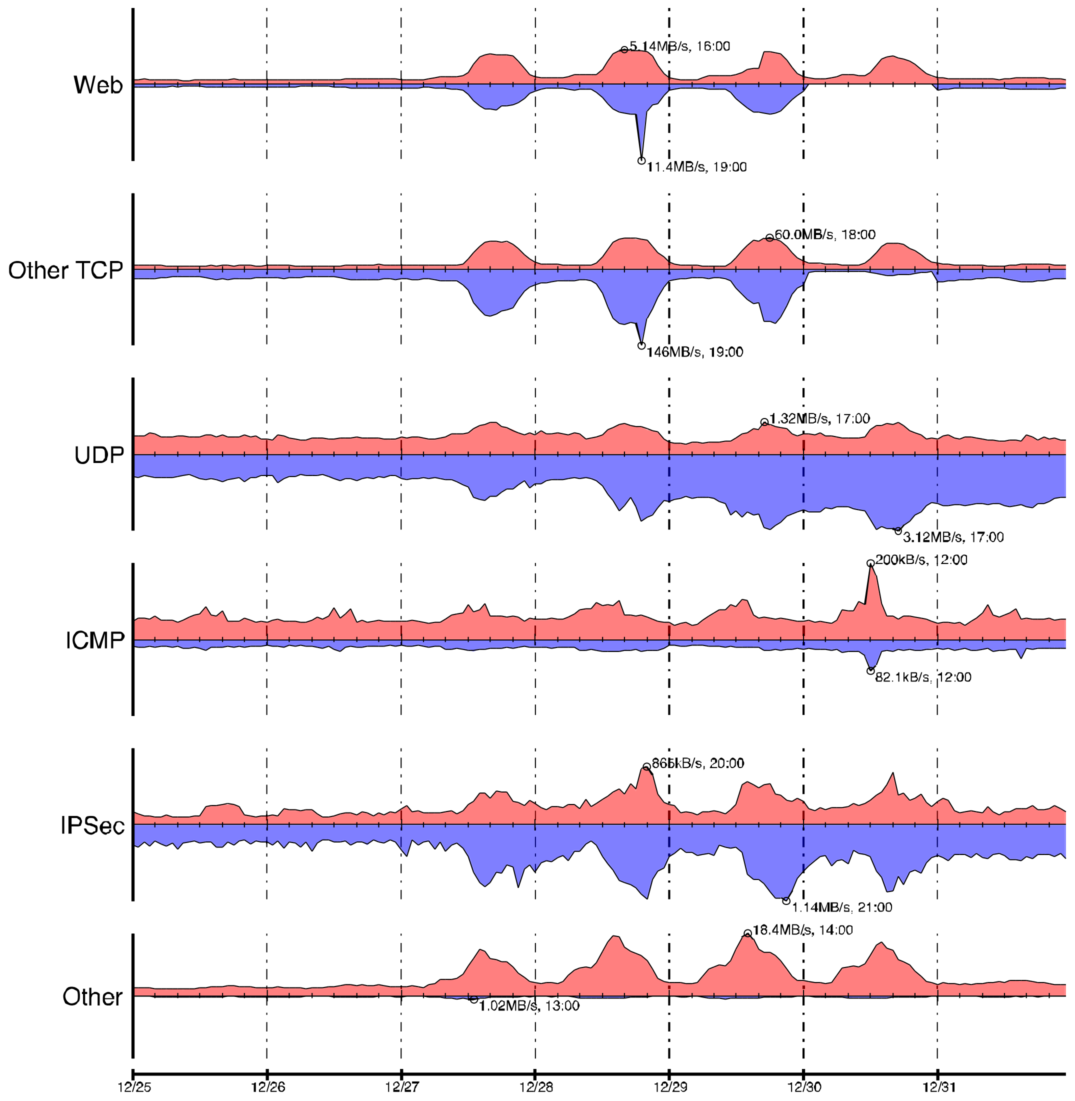


Figure 1: Multiple time series plots generated with Rayon, showing network traffic volume broken down by application and protocol. Incoming traffic is red, and outgoing traffic is blue. All the plots share the same time and volume scales, so volume can be compared across them.

Once a Plot has been created, it may be used to generate a graphic without having to understand how the Plot works. It can also be used within a larger Plot, again without having any additional knowledge. In this way, visualization developers can share and reuse their work. Figure 1 is a simple example of this. Different data are applied repeatedly to the same plot; the resulting graphics are tiled vertically to display an overall characterization of network traffic.

Engineers developing applications with visualization capabilities can write software that generates graphics from these visualization specifications knowing only the type of data it requires for input. If a visualization developer has defined events that the Plot handles, developers of interactive GUI applications can attach event listeners to them and provide interactive features. For instance, when an application end-user moves a mouse over a point in a scatterplot, the application can display a transient window containing additional information about the point.

Organizations benefit from Rayon by using or writing their own Rayon-enabled applications. Rayon comes with a set of UNIX command-line visualization tools. These tools may be used with arbitrary data, but they are designed to integrate well with the System for Internet Level Knowledge [7] (SiLK) network flow analysis tools. Each Rayon command-line tool generates a visualization based on data from a file (or UNIX standard input) and configuration information passed as command-line options. Integration is also planned with iSiLK [8], a graphical interface to the SiLK tools.

At minimum, Rayon requires a Python interpreter and one of the low-level libraries Rayon uses to generate graphics. When a dependency does not exist on a system, Rayon will, whenever possible, install with diminished capability rather than fail to install.

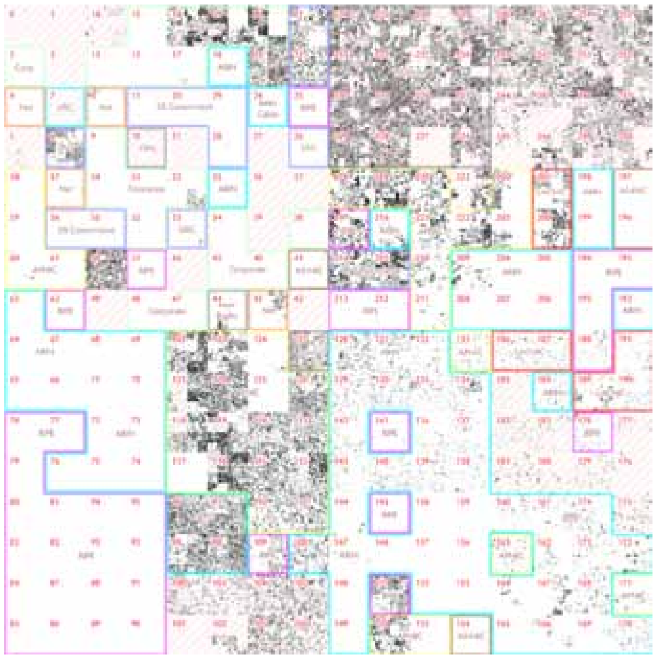


Figure 2: A visualization of network traffic generated using ryhilbert, a Rayon-enabled visualization tool that displays blocks of network addresses such that contiguous addresses are adjacent. Each point is a network block; blocks are shaded to indicate traffic volume, and colored borders are laid over the visualization to indicate IP allocations. Larger squares indicate larger networks; the red numbers in the "upper left" of blocks representing Class A networks show the first octet of those networks' addresses.

Expected Benefits

Rayon provides a visualization capability that can be used in applications across an organization. Visualization developers can create new visualizations which can generate graphics from any Rayon-enabled application, and support many different graphical use cases with a single codebase. Application developers can use the same visualization capability to support simple basic command-line tools, analysis scripts, and full-featured GUI and web applications.

2009 Accomplishments

Rayon has been incorporated into the most recent revision of an analytic tool that displays network traffic trends over time. One of the products of this tool is the visualization in Figure 1. The tool classifies network traffic by type and direction and presents the results as a set of parallel time-series graphics. The graphics share scales to make comparison easier.

Rayon has been used to implement two command-line tools. *ryscatterplot* generates scatterplots from the command line. *ryhilbert* generates graphics that represent IP networks in two-dimensional space, such that contiguous network addresses are adjacent to each other. An example is given in Figure 2.

2010 Plans

The focus of Rayon development for 2010 is to expand the number of Rayon-enabled applications. Rayon will be integrated into additional network analysis tools; one example is a tool characterizing network traffic from a set of IP addresses as using time series plots and sparklines. Command-line tools are planned to provide the user with access to time-series plots, bar plots, and line plots. Other command-line tools will be developed to perform simple analytics with obvious visual products, such as cumulative distribution functions.

In addition to expanded application support, the library will be expanded to provide additional plot types, such as matrix scatterplots and heat maps. To further support consistent styling across applications, a configuration interface is planned so that a user can provide custom styling information from a single source (e.g., a configuration file) to any Rayon-enabled application.

References

[1] The R Project for Statistical Computing. http://www.r-project.org/

[2] The MathWorks – MATLAB and Simulink for Technical Computing. http://www.mathworks.com

[3] Excel Home Page – Microsoft Office Online. http://office.microsoft.com/en-us/excel/default.aspx

[4] Python Programming Language -- Official Website. http://python.org/

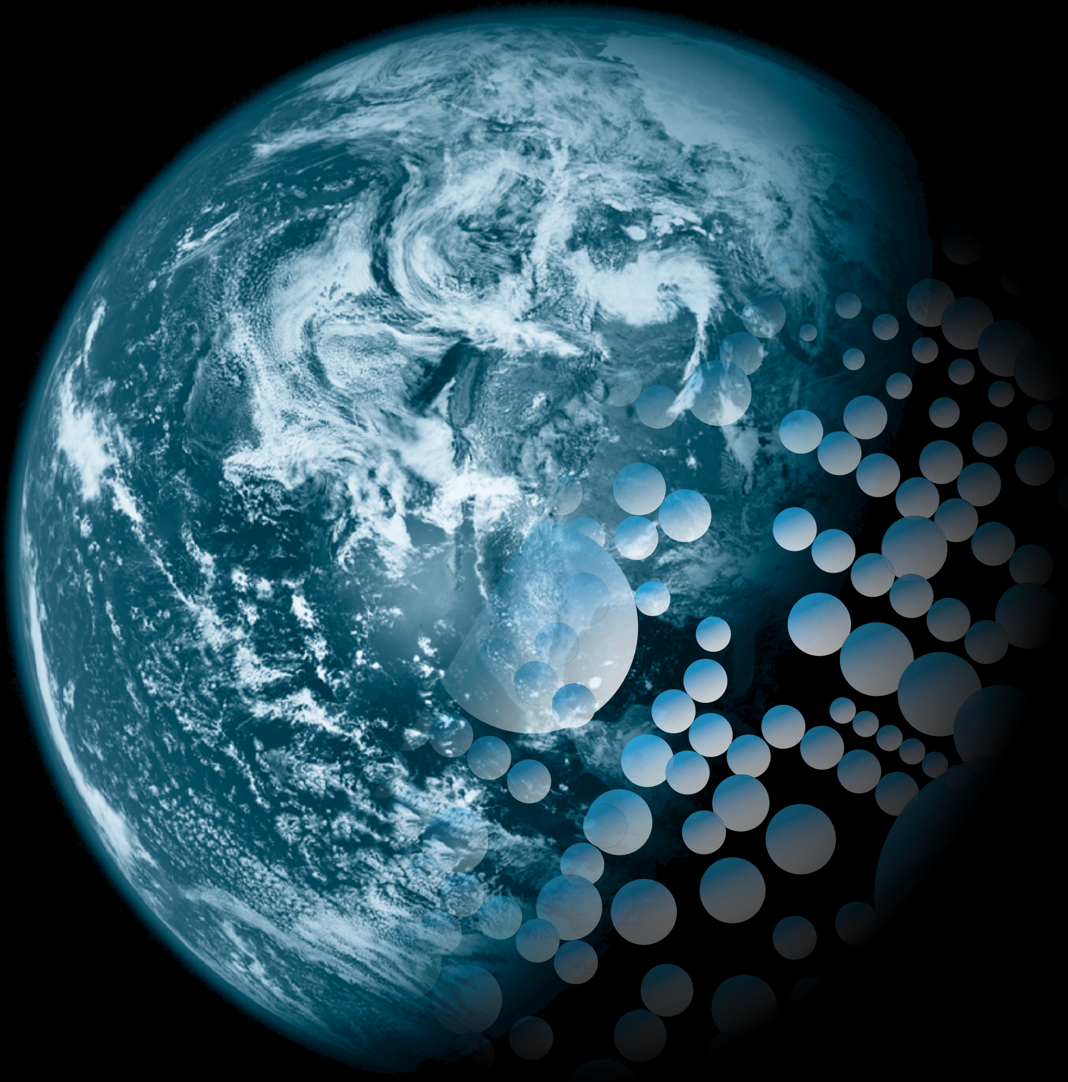
[5] Cairo. http://www.cairographics.org/

[6] wxWidgets: Cross-Platform GUI Library. http://www.wxwidgets.org/

[7] SiLK. http://tools.netsa.cert.org/silk/

[8] iSiLK. http://tools.netsa.cert.org/isilk/

Source Code Analysis Laboratory



Robert Seacord

David Svoboda

Philip Miller

Source Code Analysis Laboratory

Problem Addressed

Software vulnerability reports continue to grow at an alarming rate, with a significant number of these reports resulting in technical security alerts. To address this growing threat to governments, corporations, educational institutions, and individuals, systems must be developed that are free from software vulnerabilities.

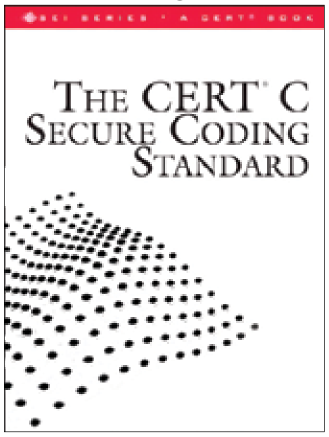
Research Approach

CERT established the Source Code Analysis Laboratory (SCALe) to address the problem of insecure code, that is, code that contains known vulnerabilities. The purpose of SCALe is to investigate approaches to developing systems that are certifiably free from known vulnerabilities.

SCALe takes a comprehensive approach to eliminating vulnerabilities and other software defects from code through these projects and tools:

1. Secure Coding Standards: Providing a detailed enumeration of coding errors that have resulted in vulnerabilities and their mitigations though the development of secure coding standards for the most commonly used software development languages.
2. Standards Development: Participating in the development and evolution of international programming language standards to improve the safety and security of common programming languages.
3. Automated Analysis Tools: Working with industry to develop tools that assist developers in building secure software.
4. CERT C and C++: Producing safe and secure executables that are known to be free from several important classes of vulnerabilities, including buffer overflows.
5. Application Conformance Testing: Offering formal certification of software as conforming to secure coding standards.
6. TSP-Secure: Integrating secure coding techniques into the Team Software Process so that high-quality, secure software can be developed with predictable cost and schedule.
7. Books, Courses, Training, and Education: Creating books and courses that create a security mindset and teach developers to code securely.

Secure Coding Standards



The SCALe foundation rests upon secure coding standards for common programming languages such as C, C++, and Java. These coding standards define an enforceable set of guidelines against which the CERT SCALe can evaluate conformance.

Coding standards encourage programmers to follow a uniform set of rules and guidelines determined by the requirements of the project and organization, rather than by the programmer's familiarity or preference. Developers and software designers can apply these coding standards during software development to create secure systems.

The use of secure coding standards defines a set of rules and recommendations to which the source code can be evaluated for conformance. Secure coding standards provide a metric for evaluating and contrasting software security, safety, reliability, and related properties.

CERT coordinates development of secure coding standards by security researchers, language experts, and software developers using a wiki-based community process. More than 450 contributors and reviewers participated in the development of secure coding standards on the CERT Secure Coding Standards wiki.

Standards Development

CERT participates in the development of international standards for programming languages to improve the safety and security of these languages. CERT is a voting member of INCITS PL22 Programming Languages, PL22.11 Programming Language C and sends technical experts to ISO/IEC working group meetings for C, C++, and programming language vulnerabilities.

Working with technical experts in these international standards bodies has led to the following advancements:

- the publication of TR 24731-1 [1] and TR 24732-2 [2], followed by their inclusion into a conditionally normative annex for C1X
- security improvements to C standard library functions
- deprecating the gets() function in C99 and removing it from C1X
- the inclusion of the Analyzability Annex into the conditionally normative annex for C1X [3]
- successful balloting of PDTR 24772.2, *Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use* [4]
- formation of the C Secure Coding Guidelines Study Group within WG14 to study the problem of producing analyzable secure coding guidelines for C99 and C1X

CERT participation in international standards bodies improves the quality of our secure coding standards and processes and provides a channel for their adoption and publication as international standards.

Automated Analysis Tools

Secure coding standards alone are inadequate to ensure secure software development because they may not be consistently and correctly applied. Manual security code audits can be supplemented through the use of automated analysis tools, including static analysis tools, dynamic analysis tools, and tools within a compiler suite. However, there are many problems and limitations in source code analysis. Static analysis techniques, while effective, are prone to both false positives and false negatives. For example, a recent study [5] found that not one of five C and C++ source analysis tools was able to diagnose 41.5% of 210 test cases, while only 7.2% of test cases were successfully diagnosed by all five tools. A similar study showed that not one of six Java code analysis tools was able to diagnose 39.7% of 177 test cases, while 0% of the test cases were discovered by all six tools. Dynamic analysis tools, while producing lower false positives rates, are prone to false negatives along untested code paths. The NIST Static Analysis Tool Exposition (SATE) also demonstrated that developing comprehensive analysis criteria for static analysis tools is difficult [6] because there are many different perspectives on what constitutes a true or false positive.

To address these problems, CERT is working with analyzer vendors and with the WG14 CSCG SG to precisely define a set of analyzable secure coding guidelines for C99, as well as for the emerging C1X major revision. Having such a set of guidelines and standardizing them through the ISO/IEC process should eliminate many of the problems encountered at NIST SATE and also improve the percentage of defects found by more than one tool. In addition to developing a set of analyzable secure coding guidelines, CERT is coordinating a test suite under a BSD-type license that will be freely available for any use. This test suite can then be used to determine which tools are capable of enforcing which guidelines and to establish false positive and false negative rates. Depending on the application, consumers of these tools may have different preferences for tools that can, for example, trade off a high false positive rate for a low false negative rate or vice versa.

In addition to working with commercial analyzer vendors, CERT has extended the Compass/ROSE tool (developed at Lawrence Livermore National Laboratory) to diagnose violations of the CERT Secure Coding Standards in C and C++ language programs. CERT has also developed a GCC prototype of the as-if infinitely ranged integer model [7] that, when combined with fuzz testing, can be used to discover integer overflow and truncation vulnerabilities.

When possible, the SCALe incorporates dynamic analysis and fuzz testing techniques in addition to the static analysis to identify coding defects and for true/false positive analysis.

CERT C and C++

Static analysis tools can be used to detect security flaws during testing and maintenance that can result in vulnerabilities. These products provide some assistance with preventing buffer overflows, but none provide certification that *all* buffer overflows are detected and prevented. However, these products do much *more* than check for buffer overflows; they detect bugs, catch other security problems, and enforce coding standards. For example, LDRA, as well as other static analysis tool vendors, have implemented code checkers to diagnose violations of these guidelines in C and C++ source code.

For any solution to make a significant difference in the reliability of the software infrastructure, the methods must be incorporated into tools that working programmers are using to build their applications.

Compiler producers constitute a segment of the software production supply chain, one that is quite different from the quality-tools producers. Each hardware company typically maintains some number of compiler groups, as do several of the large software producers. There are several specialized compiler producers. In addition, there is a significant community of individuals and companies that support the open-source GCC. Adding these various groups together, we estimate that there are well over 100 compiler vendors.

Safe Secure C/C++ (SSCC) eliminates several important classes of vulnerabilities, including writing outside the bounds of an object (e.g., buffer overflow), reading outside the bounds of an object, and arbitrary reads/writes (e.g., wild-pointer stores) [8]. The buffer overflow problem, for example, is solved using static analysis for issues that can be resolved at compile and link time and dynamic analysis using highly optimized code sequences for issues that can be resolved only at run time.

To encourage adoption of the SSCC methods into working compilers, CERT proposes extending ROSE to perform the SSCC analysis methods and produce an *advice* file for the platform-dependent compiler. The structure of such a tool was first proposed as a method for providing optimization advice from a front-end source analysis tool to a platform-dependent *back-end* compiler [9]. For example, the modified ROSE tool can advise the back-end compilers to "verify that pointer p is less than a+1000, just before the expression at line 21 token 7" or "insert 5 elements of padding after the array a declared at line 31 token 3."

Along with the ROSE Advisor, a *pre-linker* is also required to read and process the full collection of bounds-data files from all components of the application being compiled and linked. The resulting system is illustrated in Figure 1.

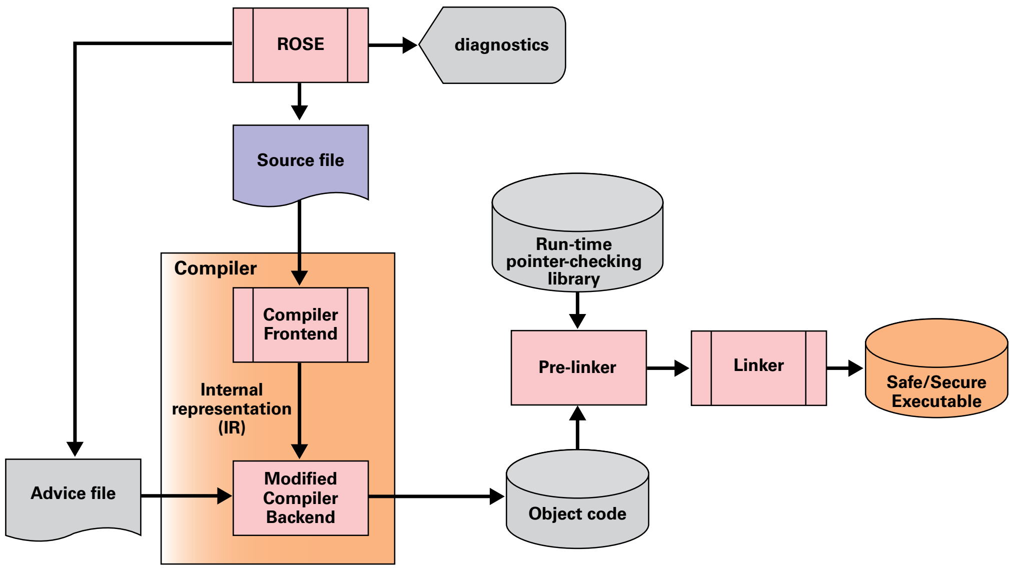


Figure 1: ROSE Advisor design

Application Conformance Testing

SCALe provides an operational capability for application conformance testing to satisfy the demand for source code assessments for government and industry. CERT uses the SCALe to assess client source code against one or more secure coding standards, following the process shown in Figure 2. A detailed report of findings is provided to the customer to repair. After these findings have been addressed by the developer, SCALe issues and certifies the conformance test results.

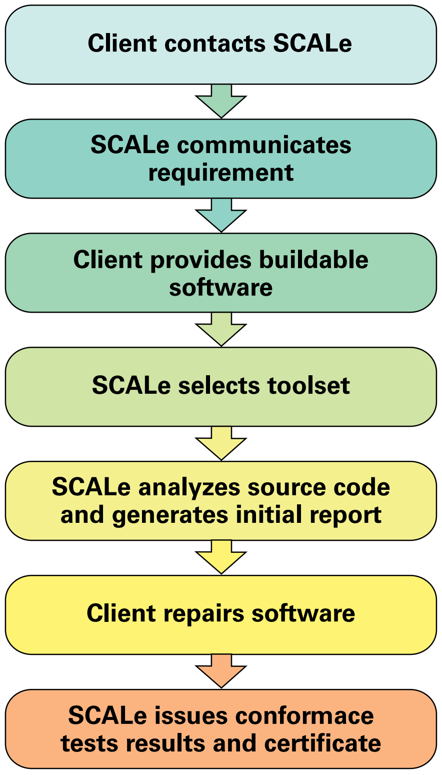

Client contacts SCALe
SCALe communicates requirement
Client provides buildable software
SCALe selects toolset
SCALe analyzes source code and generates initial report
Client repairs software
SCALe issues conformace tests results and certificate

Figure 2: Conformance testing process

For each rule and recommendation, the source code is certified as provably nonconforming, deviating, conforming, or provably conforming.

- The code is provably nonconforming if one or more violations of a rule are discovered for which no deviation has been specified.
- Deviating code is code for which the application developer has a documented deviation. This documentation is included with the certification.
- The code is conforming if no violations of a rule could be identified.
- Finally, the code is provably conforming if the code has been verified to adhere to the rule in all possible cases.

Once the process is completed, a report detailing the conformance or nonconformance for each CERT C Secure Coding rule is provided to the customer.

TSP-Secure

The SEI Team Software Process^{SM} (TSP^{SM}) methodology, known for enabling dramatic improvement in productivity and product quality, is now being used for rapid, economic, and self-sustaining CMMI implementation. TSP-Secure extends TSP to achieve the development of secure software systems by institutionalizing guidance offered from CERT, as illustrated in Figure 3. By implementing TSP-Secure, organizations can efficiently build high-quality, secure software while conforming to CMMI.

TSP-Secure incorporates the planning, process, quality, measurement, and tracking frameworks of TSP for secure software development and generates the practices and artifacts required to satisfy a CMMI SCAMPI Maturity Level 3 (ML3) appraisal. TSP-Secure requires selection of one or more secure coding standards during the requirements phase of the project. TSP-Secure teams apply the application conformance testing processes as part of their own development processes to produce demonstrably conforming secure code.

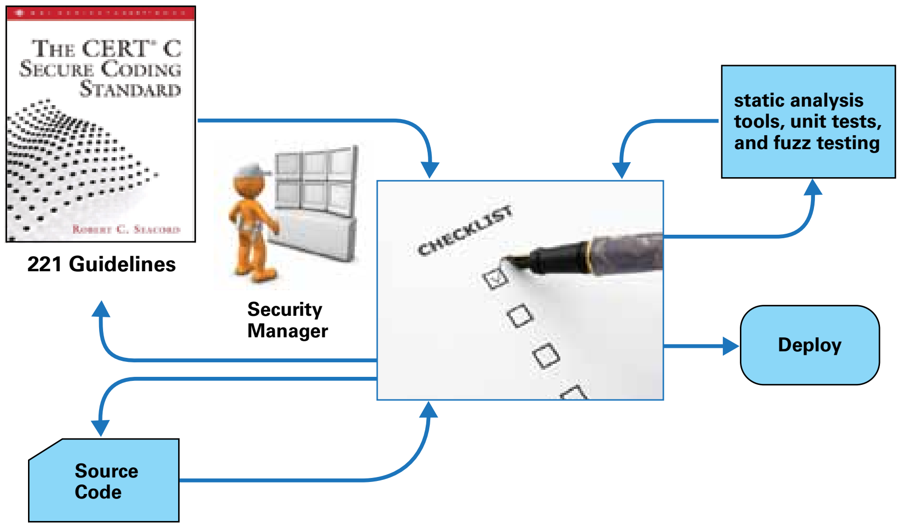

The CERT C Secure Coding Standard
Robert C. Seacord
221 Guidelines
Security Manager
CHECKLIST
static analysis tools, unit tests, and fuzz testing
Deploy
Source Code

Figure 3: TSP-Secure

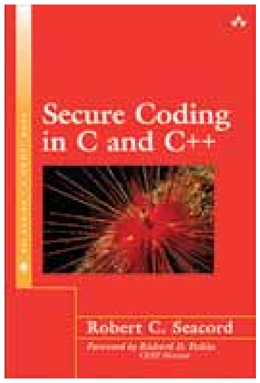
Technical training for developers is delivered prior to project launch. A new team role, Security Manager, is defined.

Additional launch meetings are specified and scripted. Some existing launch meetings are modified. These include modified scripts and forms. Process steps integrate the use of static analysis tools and so forth. At this time, development teams must be using C or C++ to take advantage of the security training, tools, and methods. We expect to extend that to Java development in the coming year.

Finally, feedback loops exist for putting fresh information discovered in conducting TSP-Secure projects back into our security and information repositories.

Books, Courses, Training, and Education

CERT has published two books on secure coding: *Secure Coding in C and C++* [10] and *The CERT C Secure Coding Standard* [11]. These books identify insecure coding practices, describe how insecure code can be exploited, and provide mitigation strategies.



CERT has developed a four-day *Secure Coding in C and C++* course that identifies common programming errors in C and C++ and describes how these errors can lead to code that is vulnerable to exploitation. The course concentrates on security issues intrinsic to the C and C++ programming languages and associated libraries and is based on the Addison-Wesley book by the same name. This course is currently being offered by the SEI and by SEI partner organizations.

CERT is also involved in teaching secure programming to undergraduates in the Computer Science department at Carnegie Mellon and secure software engineering to graduate students in Carnegie Mellon's Information Networking Institute, and is working with other universities to improve their software security courses.

Expected Benefits

The goal of the CERT SCALe is to reduce or eliminate vulnerabilities deployed in operational software by preventing coding errors or discovering and eliminating security flaws during implementation and test. Organizations can benefit from this work by

- participating in the development of CERT Secure Coding Standards and applying these standards in their software development process
- adopting, extending, and using static analysis tools (some of which are freely available) that have been enhanced to detect violations of CERT Secure Coding guidelines
- training their software development workforce through secure coding courses developed and offered by the SEI and partner organizations
- using the resources of the CERT SCALe for conformance testing and possible certification
- using TSP-Secure as their software development process

2009 Accomplishments

As-if Infinitely Ranged Integer Model

In 2009, CERT developed the as-if infinitely ranged (AIR) integer model, which provides a largely automated mechanism for eliminating integer overflow and integer truncation [7]. The AIR integer model either produces a value equivalent to one that would have been obtained using infinitely ranged integers or results in a runtime constraint violation. A proof of concept modification to GCC compiler version 4.5.0

has been developed by CERT that automatically invokes a runtime constraint handler when an integral operation fails to produce a correctly represented value.

Secure Design Patterns

CERT has completed a two-year joint study with JPCERT, Japan's first computer security incident response team (CSIRT), to develop a technical report describing secure design patterns. Secure design patterns are descriptions or templates describing a general solution to a security problem that can be applied in many different situations. Rather than focus on the implementation of specific security mechanisms, the secure design patterns detailed in the report are meant to eliminate the accidental insertion of vulnerabilities into code or to mitigate the consequences of vulnerabilities. The patterns were derived by generalizing existing best security design practices and by extending existing design patterns with security-specific functionality. They are categorized according to their level of abstraction: architecture, design, or implementation. Six new secure design patterns were added to the report in an October 2009 update [12].

International Standards

CERT is working with both the ISO/IEC JTC1/SC22/WG14 international standardization working group for the programming language C and the WG21 international standardization working group for the programming language C++ to improve security in these languages. CERT is also working with the WG23 Programming Language Vulnerabilities working group in the development of TR 24772, *Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use.*

In 2009, the ISO/IEC JTC 1/SC 22/WG 14 international standardization working group for the programming language C has created the C Secure Coding Guidelines Study Group (CSCG SG) to study the problem of producing analyzable secure coding guidelines for C99 and C1X.

CERT also provides the liaison from the PL22 Programming Languages to CS1 (cyber security). CERT is also providing expertise to the JTC 1/SC 7 embedded system coding guidelines study group.

SCALe Assessments

CERT conducted software security assessments for AREVA and the Office of Navy Intelligence. These assessments included systems developed in C, C++, and Java.

2010 Plans

International Standards Activities

CERT is hosting the March 2010 meeting of the ISO/IEC JTC1/SC22/WG21 international standardization working group for the programming language C++ on the Carnegie Mellon University campus in Pittsburgh, Pennsylvania.

Technical experts from CERT will continue to work with international standards bodies to improve the security of programming languages.

Secure Coding Standards=

CERT Secure Coding Initiative will continue to collaborate with Sun Microsystems to develop The CERT Sun Microsystems Secure Coding Standard for Java [13]. This standard provides guidance for secure programming in the Java Platform Standard Edition 6 environment. Programmers who adopt the Java standard can avoid vulnerabilities in Java systems. This coding standard affects the wide range of products coded in Java, such as PCs, game players, mobile phones, home appliances, and automotive electronics.

CERT will also continue to develop a C++ Secure Coding Standard [14] and maintain and enhance the existing C Secure Coding Standard.

Application Conformance Testing

During 2010 CERT plans to offer conformance testing against the CERT C Secure Coding Standard to any interested parties. We are also hoping to certify a conforming system and document the process in a case study.

References

[1] ISO/IEC. *Extensions to the C Library, — Part I: Bounds-checking* interfaces (ISO/IEC TR 24731-1). Geneva, Switzerland: International Organization for Standardization, April 2006.

[2] ISO/IEC. *Extensions to the C Library, — Part II: Dynamic Allocation Functions* (ISO/IEC PDTR 24731-2). Geneva, Switzerland: International Organization for Standardization, August 2007.

[3] Plum, T. & Seacord, R. C. "ISO/IEC JTC 1/SC 22/WG14/N1394 – Analyzability." Geneva, Switzerland: International Standards Organization, August 2009. http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1394.pdf

[4] ISO/IEC. *Information Technology — Programming Languages — Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use* (ISO/IEC PDTR 24772). Geneva, Switzerland: International Standards Organization, March 2008. http://www.aitcnet.org/isai/DocLog/180-thru-199/22-WG23-N-0191/n0191.pdf

[5] Landwehr, C. IARPA STONESOUP Proposers Day. July 2008. http://www.iarpa.[6] Okun, V., Gaucher, R., & Black, P. E. *Static Analysis Tool Exposition (SATE) 2008*. NIST Special Publication 500-279. June 2009.

[7] Keaton, D., Plum, T., Seacord, R. C., Svoboda, D., Volkovitsky, A., & Wilson, T. *As-if Infinitely Ranged Integer Model* (CMU/SEI-2009-TN-023). Carnegie Mellon University, Software Engineering Institute, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tn023.cfm

[8] Plum, T. & Keaton, D. M. "Eliminating Buffer Overflows Using the Compiler or a Standalone Tool." *Proceedings of the Workshop on Software Security Assurance Tools, Techniques, and Metrics*. Long Beach, California, November 7-8, 2005. https://samate.nist.gov/index.php/Past_Workshops

[9] Sangyeun, C., Tsaiy, J-Y., Song, Y., Zheng, B., Schwinn, S. J., Wang, X., Zhao, Q., Liz, Z., Lilja, D. J., & Yew, P-C. High-Level Information – An Approach for Integrating Front-End and Back-End Compilers. http://www.cs.pitt.edu/~cho/data/cho-icpp98.pdf

[10] Seacord, R. C. *Secure Coding in C and C++*. Addison-Wesley Professional, 2005 (ISBN: 0-321-33572-4).

[11] Seacord, R. C. *The CERT C Secure Coding Standard*. Addison-Wesley Professional, 2008 (ISBN: 0-321-56321-2).

[12] Dougherty, C., Sayre, K., Seacord, R. C., Svoboda, D., & Togashi, K. *Secure Design Patterns* (CMU/SEI-2009-TR-010). Carnegie Mellon University, Software Engineering Institute, 2009. http://www.sei.cmu.edu/reports/09tr010.pdf

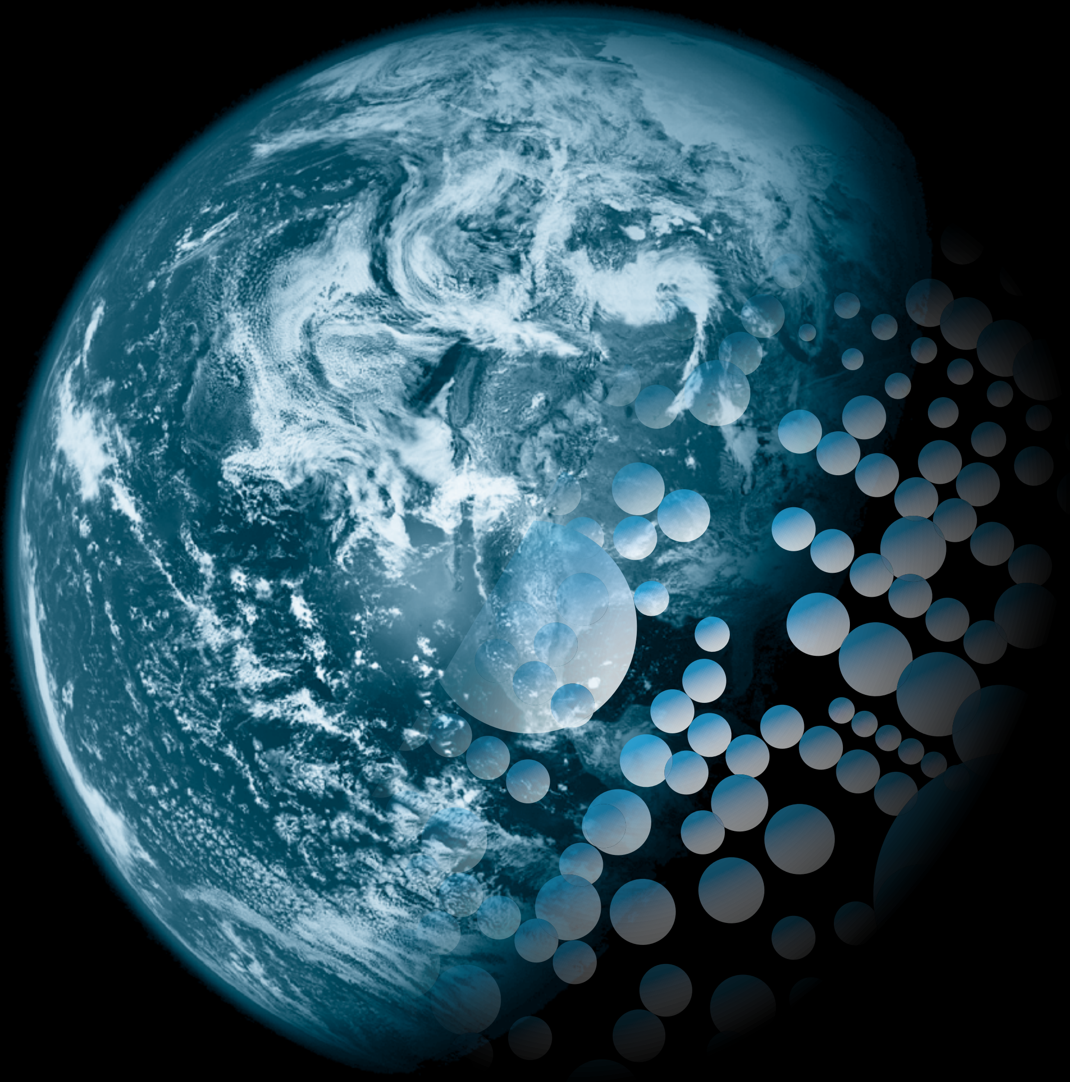
[13] The CERT Sun Microsystems Secure Coding Standard for Java. Pittsburgh, PA: Software Engineering Institute, CERT, 2009. https://www.securecoding.cert.org/confluence/x/Ux

[14] The CERT C++ Secure Coding Standard. Pittsburgh, PA: Software Engineering Institute, CERT, 2009. https://www.securecoding.cert.org/confluence/x/fQI

Nancy R. Mead

Justin Zahn

SQUARE: Requirements Engineering for Improved System Security



SQUARE: Requirements Engineering for Improved System Security

Problem Addressed

It is well recognized in industry that requirements engineering is critical to the success of any major development project. Several authoritative studies have shown that requirements engineering defects cost 10 to 200 times as much to correct once fielded than if they are detected during requirements development [1, 2]. A more recent vendor example indicates that it is 100 times cheaper to fix security flaws at requirements time than after a product has been released [3]. Other studies have shown that reworking requirements, design, and code defects on most software development projects costs 40 to 50 percent of total project effort [4], and the percentage of defects originating during requirements engineering is estimated at more than 50 percent [5]. The total percentage of project budget due to requirements defects is 25 to 40 percent [6].

The National Institute of Standards and Technology (NIST) reports that software faulty in security and reliability costs the economy $59.5 billion annually in breakdowns and repairs [7]. The costs of poor security requirements make apparent that even a small improvement in this area will provide a high value. By the time an application is fielded and in its operational environment, it is very difficult and expensive to significantly improve its security.

Requirements problems are among the top causes of why

- projects are significantly over budget, past schedule, have significantly reduced scope, or are cancelled
- development teams deliver poor-quality applications
- products are not significantly used once delivered

These days we have the further problem that the environment in which we do requirements engineering has changed, resulting in an added element of complexity. Software development occurs in a dynamic environment that changes while projects are still in development, with the result that requirements are in flux from the beginning. This can be due to conflicts between stakeholder groups, rapidly evolving markets, the impact of tradeoff decisions, and so on.

When security requirements are considered at all during the system life cycle, they tend to be general lists of security features such as password protection, firewalls, virus detection tools, and the like. These are, in fact, not security requirements at all but rather implementation mechanisms that are intended to satisfy unstated requirements, such as authenticated access. As a result, security requirements that are specific to the system and that provide for protection of essential services and assets are often neglected. In addition, the attacker perspective is not considered, with the result that security requirements, when they exist, are likely to be incomplete. We believe that a systematic approach to security requirements engineering will help to avoid the problem of generic lists of features and to take into account the attacker perspective.

In reviewing requirements documents, we typically find that security requirements, when they exist, are in a section by themselves and have been copied from a generic set of security requirements. The requirements elicitation and analysis that is needed to get a better set of security requirements seldom takes place.

Much requirements engineering research and practice has addressed the capabilities that the system will provide. So while significant attention is given to the functionality of the system from the user's perspective, little attention is given to what the system should *not* do. In one discussion on requirements prioritization for a specific large system, ease of use was assigned a higher priority than security requirements. Security requirements were in the lower half of the prioritized requirements. This occurred in part because the only security requirements that were considered had to do with access control.

Research Approach

The CERT Program has developed a methodology to help organizations build security into the early stages of the production life cycle. The Security Quality Requirements Engineering (SQUARE) methodology consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. Although SQUARE could likely be generalized to any large-scale design project, it was designed for use with information technology systems.

The SQUARE process is most effective and accurate when it is conducted by a team of requirements engineers with security expertise and the stakeholders of an IT project. It begins with the requirements engineering team and project stakeholders agreeing on technical definitions that serve as a baseline for all future communication. Next, assets are identified and business and security goals are outlined. Third, artifacts and documentation are created, which are necessary for a full understanding of the relevant system. A structured risk assessment determines the likelihood and impact of possible threats to the system.

Following this work, the requirements engineering team determines the best method for eliciting initial security requirements from stakeholders. This determination depends on several factors, including the stakeholders involved, the expertise of the requirements engineering team, and the size and complexity of the project. Once a method has been established, the participants rely on artifacts and risk assessment results to elicit an initial set of security requirements. Two subsequent stages are spent categorizing and prioritizing these requirements for management's use in making tradeoff decisions. Finally, an inspection stage is included to ensure the consistency and accuracy of the security requirements that have been generated.

SQUARE's nine discrete steps are outlined in Table 1. Each step identifies the necessary inputs, major participants, suggested techniques, and final output. Generally, the output of each step serves as the sequential input to the ensuing steps, though some steps may be performed in parallel. For instance, it might be more efficient for the requirements engineering team to perform Step 2 (Identify Assets and Security Goals) and Step 3 (Develop Artifacts) simultaneously, since to some extent they are independent activities. The output of both steps, however, is required for Step 4 (Perform Risk Assessment). In principle, Steps 1–4 are actually activities that precede security requirements engineering but are necessary to ensure that it is successful.

The SQUARE process is described in a technical report [8] and is suitable for incorporation into development practice. SQUARE is described in the Requirements Engineering section of the Build Security In website [9] and in three books [10, 11, 12]. CERT is currently continuing research and application of the process and is working to develop a robust tool to support each stage of the methodology.

Expected Benefits

When SQUARE is applied, the user should expect to have identified, documented, and inspected relevant security requirements for the system or software that is being developed. SQUARE may be more suited to a system under development or one undergoing major modification than one that has already been fielded, although it has been used both ways.

2009 Accomplishments

In conjunction with the Carnegie Mellon University CyLab and Master of Software Engineering Program, an MSE Studio team has developed a robust SQUARE tool that is available for free. A workshop on SQUARE was delivered to the National Defense University of Taiwan, and the workshop materials were translated into Chinese and delivered there. Research on privacy requirements engineering with SQUARE (P-SQUARE) has continued, and there have been several published papers [13, 14, 15]. A white paper on SQUARE for Acquisition (A-SQUARE) has been produced.

2010 Plans

Carnegie Mellon University student interns and volunteers in the community are working to extend SQUARE for privacy. The extensions of SQUARE for Acquisition will be incorporated into the educational materials. In addition, we will continue to take advantage of client opportunities to apply SQUARE, P-SQUARE, and A-SQUARE in the field and will publish the results.

References

[1] Boehm, B. W. & Papaccio, P. N. "Understanding and Controlling Software Costs." *IEEE Transactions on Software Engineering SE-4*, 10 (October 1988): 1462–77.

[2] McConnell, Steve. "From the Editor - An Ounce of Prevention." *IEEE Software 18*, 3 (May 2001): 5–7.

[3] Meftah, B. "Business Software Assurance: Identifying and Reducing Software Risk in the Enterprise." https://buildsecurityin.us-cert.gov/swa/downloads/Meftah.pdf

[4] Jones, C., ed. *Tutorial: Programming Productivity: Issues for the Eighties, 2nd Ed.* IEEE Computer Society Press, 1986.

[5] Wiegers, K. E. "Inspecting Requirements" (column). *StickyMinds*, July 30, 2001. http://www.stickyminds.com

[6] Leffingwell, D. & Widrig. D. *Managing Software Requirements–A Use Case Approach*, 2nd ed. Addison-Wesley, 2003.

[7] National Institute of Standards and Technology. "Software Errors Cost U.S. Economy $59.5 Billion Annually" (NIST 2002-10). http://www.nist.gov/public_affairs/releases/n02-10.htm (2002).

[8] Mead, N. R., Hough, E., & Stehney, T. *Security Quality Requirements Engineering (SQUARE) Methodology* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html

[9] Software Engineering Institute. Build Security In. https://buildsecurityin.us-cert.gov/ (2009).

[10] Mead, N. R., Davis, N., Dougherty, C., & Mead, R. Ch. 8, "Recommended Practices," 275–308. *Secure Coding in C and C++,* Robert Seacord. Addison Wesley Professional, 2005.

[11] Mead, N. R. Ch. 3, "Identifying Security Requirements Using the SQUARE Method," 44–69. *Integrating Security and Software Engineering: Advances and Future Visions* H. Mouratidis & P. Giorgini. Idea Group, 2006 (ISBN: 1-59904-147-2).

[12] Allen, J., Barnum, S., Ellison, R., McGraw, G., & Mead, N. R. *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley Professional, 2008 (ISBN-13: 978-0-321-50917-8).

[13] Abu-Nimeh, S., Miyazaki, S., & Mead, N. R. "Integrating Privacy Requirements into Security Requirements Engineering, 542–547. *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE)*. Boston, MA, July 1–3, 2009. IEEE Computer Society, 2009.

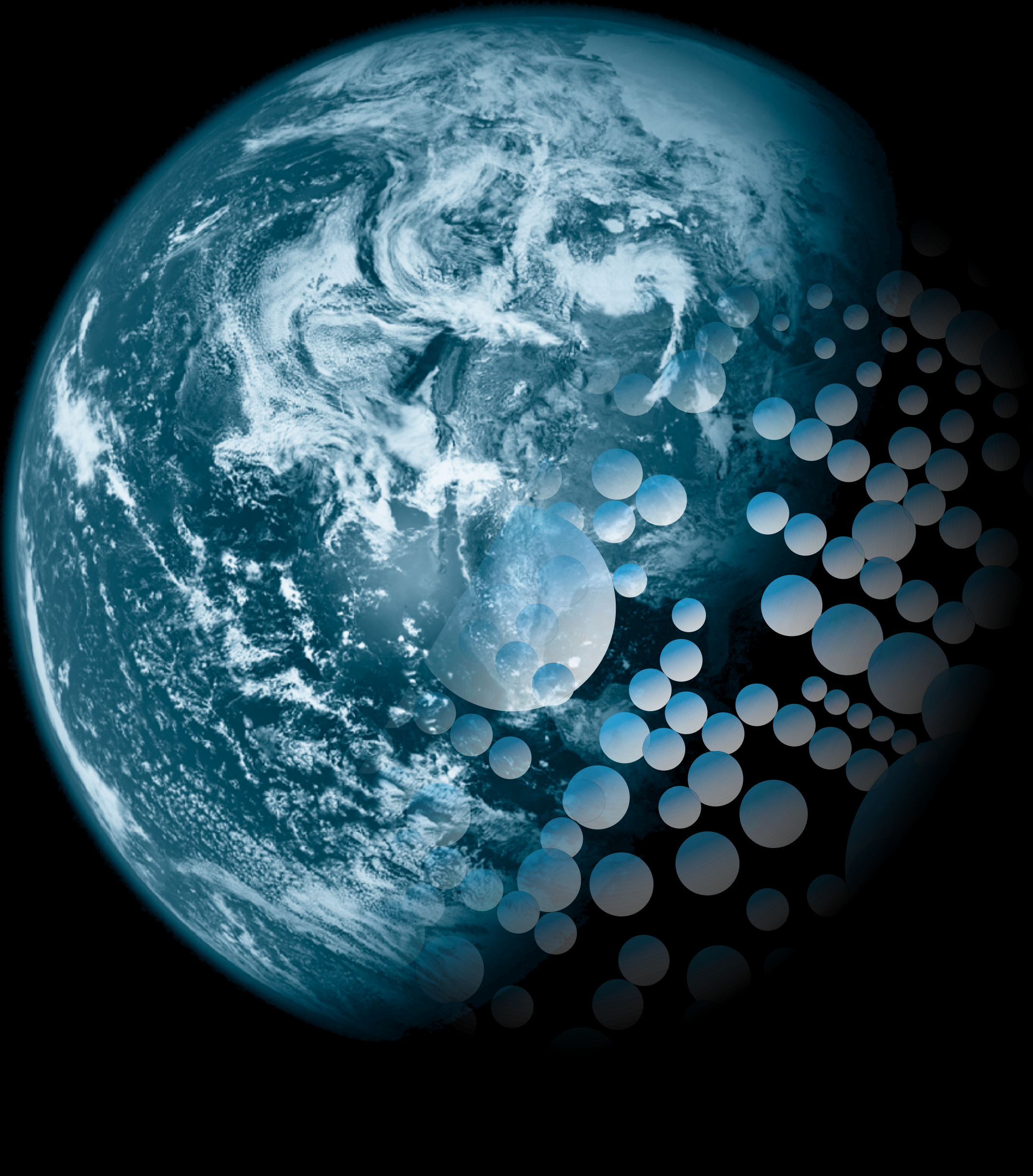
[14] Mead, N. R. & Yoshioka, N. "Square Up Your Security Requirements Engineering with SQUARE." *Information Processing Society of Japan (IPSJ) Journal* (invited) *50*, 3 (March 2009).

[15] Miyazaki, S., Mead, N. R., & Zhan, J. "Computer-Aided Privacy Requirements Elicitation Technique," 367–372. *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference (APSCC)*. Yilan, Taiwan, December 9–12, 2008. IEEE Computer Society, 2009.

Table 1: Security Requirements Elicitation and Analysis Process

	Step	Input	Techniques	Participants	Output
1	Agree on definitions	Candidate definitions from IEEE and other standards	Structured interviews, focus group	Stakeholders, requirements team	Agreed-to definitions
2	Identify assets and security goals	Definitions, candidate goals, business drivers, policies and procedures, examples	Facilitated work session, surveys, interviews	Stakeholders, requirements engineer	Assets and goals
3	Develop artifacts to support security requirements definition	Potential artifacts (e.g., scenarios, misuse cases, templates, forms)	Work session	Requirements engineer	Needed artifacts: scenarios, misuse cases, models, templates, forms
4	Perform risk assessment	Misuse cases, scenarios, security goals	Risk assessment method, analysis of anticipated risk against organizational risk tolerance, including threat analysis	Requirements engineer, risk expert, stakeholders	Risk assessment results
5	Select elicitation techniques	Goals, definitions, candidate techniques, expertise of stakeholders, organizational style, culture, level of security needed, cost benefit analysis, etc.	Work session	Requirements engineer	Selected elicitation techniques
6	Elicit security requirements	Artifacts, risk assessment results, selected techniques	Joint Application Development (JAD), interviews, surveys, model-based analysis, checklists, lists of reusable requirements types, document reviews	Stakeholders facilitated by requirements engineer	Initial cut at security requirements
7	Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints	Initial requirements, architecture	Work session using a standard set of categories	Requirements engineer, other specialists as needed	Categorized requirements
8	Prioritize requirements	Categorized requirements and risk assessment results	Prioritization methods such as Triage, Win-Win	Stakeholders facilitated by requirements engineer	Prioritized requirements
9	Inspect requirements	Prioritized requirements, candidate formal inspection technique	Inspection method such as Fagan, peer reviews	Inspection team	Initial selected requirements, documentation of decision making process and rationale

Additional Research



Advanced Technology for Test & Evaluation of Embedded Systems

Timothy Daly and Richard Linger

Embedded system test and evaluation (T&E) can consume substantial resources in system development and evolution. Yet despite significant expenditures of time and effort, T&E can nevertheless produce inconclusive results, and it is well known that even thoroughly tested software can experience errors and vulnerabilities in field use. Software systems can exhibit massive numbers of possible execution paths, and no testing effort, no matter how well planned and funded, can exercise more than a small fraction of them. The need for more effective validation of system functionality across the life cycle is discussed in the report of the Defense Science Board (DSB) Task Force on Defense Software [1]:

"Support for assurance and certification is a critical area that needs to be addressed proactively during each step of the development process. After-the-fact or post-mortem complete assurance and certification methods, needed after every change, are very expensive, and can involve rediscovery of the design decisions made in the development process. While there is certainly a key place for independent testing and certification methods, the earlier that support for assurance and certification is introduced into the development process, the lower the costs."

Test and evaluation of embedded systems can be particularly difficult because of the frequent need for specialized hardware environments and test interfaces. This exploratory project is investigating the possibility of applying advanced technology for a new approach to embedded system T&E. It involves a new form of automated software analysis that may permit reduction or replacement of certain types of testing.

The foundation for this work is CERT-developed Function Extraction (FX) technology [2]. The objective of FX is to automate computation of software behavior with mathematical precision to the maximum extent possible. Initial work relating FX to T&E [3, 4] suggests that the technology can support software verification and assurance, and may reduce or replace certain forms of testing, most notably functional testing at the unit and component level. Of particular interest is exploration of the technology for T&E in applications such as avionics, space, and weapons systems, where assurance of functional behavior and detection of malicious content are essential. This approach may permit T&E to expand beyond an end-of-life-cycle role, to address earlier phases where corrections can require fewer resources and carry less risk.

The FX approach does not require selecting specific executions among so many possibilities as is the case in testing. Behavior computation can uncover errors, vulnerabilities, and malicious content that may not be found in testing, simply because the specific combinations of conditions for their execution may not be exercised. The key observation is that if software behavior is known, functional testing may provide little additional information, and can be reduced or in some cases even eliminated. Other forms of testing will still be required, for example, performance and integration testing, but substantial savings may be possible at the functional testing level.

Figure 1 depicts differences in the testing and behavior computation approaches. The flow on the left depicts a program subjected to traditional testing. Because programs typically exhibit very large populations of possible executions, as a practical matter only a small fraction of them can be exercised in the testing process. This partial coverage is illustrated by the yellow population of executions that is not addressed by the black dots representing individual test cases. The flow on the right depicts behavior computation, which produces results not from selective program execution as in testing, but rather from comprehensive static analysis to determine program behavior. This analysis computes all possible behavior and expresses it in a set of disjointed partitions which define and cover the green behavior space in the Figure. Each partition is defined by a predicate, which, if true on input to the program, results in transformation of the input state into the output state through a set of concurrent assignments associated with that partition. These results can be analyzed for conformance to requirements and specifications, and to investigate whether malicious content or corrupted functionality is present.

This project will explore potential savings in testing resources and improvement in software quality through application of behavior computation technology.

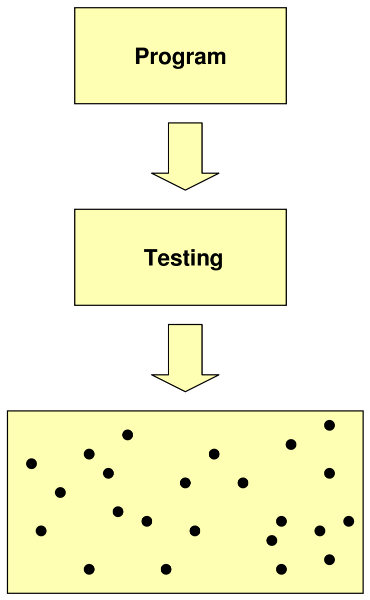
References

[1] Hansen, M. and Nesbit, R., *Report of the Defense Science Board Task Force on Defense Software,* Defense Science Board, Wshing, DC, Nov. 2000.

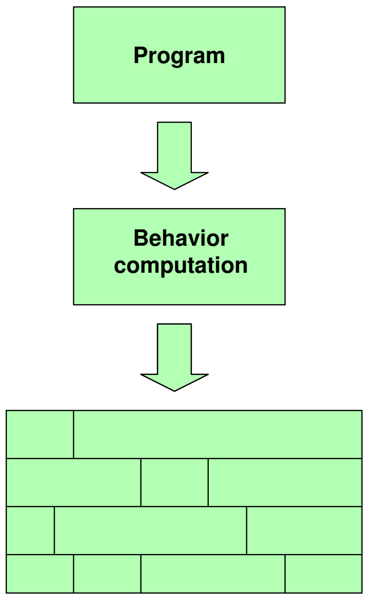
[2] Linger, R., Pleszkoch, M., Burns, L., Hevner, A., and Walton, G. (2007). "Next-Generation Software Engineering: Function Extraction for Computation of Software Behavior," *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS40)*, Hawaii, IEEE Computer Society Press, Los Alamitos, CA.

[3]Linger, R., Pleszkoch, M., and Hevner, R., "Introducing Function Extraction into Software Testing," *The Data Base for Advances in Information Systems: Special Issue on Software Systems Testing,* ACM SIGMIS, New York, NY, 2008.

[4] Bartholomew, R., Burns, L., Daly, T., Linger, R., and Prowell, S., "Function Extraction: Automated Behavior Computation for Aerospace Software Verification and Certification," *Proceedings of 2007 AIAA Aerospace Conference*, Monterey, CA, May, 2007, Vol. 3, pp. 2145-2153.



Population of possible program executions: Dots are test cases, yellow is untested executions



Population of possible program behaviors: Disjoint partitions are computed behavior cases that cover the entire population

Figure 1: Comparison of Testing and Behavior Computation for T&E

Automatic Generation of Hidden Markov Models for the Detection of Polymorphic and Metamorphic Malware

Mark Pleszkoch, Cory F. Cohen, and Timothy Daly

As of July 2009, the CERT Artifact Catalog contained nearly 7 million executable files. Accurate identification and classification of these malware artifacts is essential for gaining intellectual control over the contents of the catalog. Additionally, many subsequent analysis activities performed by CERT are critically dependant on accurate malware identification and classification. Such activities include malware trend analysis (time-evolution of the quantity and distribution of malware varieties) and large-scale analysis of specific malware families (overall analysis of an entire group or family of malware). Since CERT stakeholders require malware trending information to support strategic planning, accuracy in carrying out these activities is essential.

Over half a dozen existing anti-virus products are regularly run over the contents of the CERT artifact catalog. However, these anti-virus products are unable to meet the need of accurately classifying executable malware artifacts for several reasons, chief among them being the frequency with which they yield different and conflicting classifications. This is likely due to several factors, including use of different techniques and criteria for malware detection, coupled with the fact that malware classification is a secondary concern for those products, which are primarily focused on separation of malicious from non-malicious code.

Because of this, CERT researchers developed the Pithos tool [1] to cluster and classify malware artifacts based solely on their entry point sequence, that is, the first 100 bytes of code or data starting from the entry point. Pithos works extremely well in identifying most malware families, but performs sub-optimally on *polymorphic* and *metamorphic* malware. Such malware is constructed from the output of a code generation engine that randomly produces varying Intel code sequences to accomplish the same desired behavior. Because of this, Pithos signatures, which match specific bytes (or parts of bytes) at specific offsets, are only created when the same unique sequence of bytes is generated at the same offset in enough different instances of the polymorphic or metamorphic malware to reach the Pithos signature creation threshold.

To address this problem, CERT researchers first investigated the Allaple polymorphic malware family, which accounts for over 3% of the entries in the artifact catalog. Instances of Allaple are surrounded by two layers of polymorphic packing, so that the bytes at the outer layer have both randomized instructions and randomized data. This makes detection of Allaple more difficult, and helps explain why Allaple is so prevalent in the artifact catalog. However, by examining known instances of Allaple, it was possible to develop a state machine model of Allaple's code generation engine. An Allaple detector was then implemented by using the state machine model as a pattern against which to match the entry point sequences of prospective artifacts. This detector proved to be highly accurate, exhibiting the ability to correctly identify instances of Allaple after examining only 25 bytes of the entry point sequence. More importantly, Pithos performed even better on the remaining collection of artifacts after all the Allaple instances were first removed.

The results of the Allaple investigation indicated that malware identification for the artifact catalog would significantly benefit from application of state machine models for each of the known polymorphic and metamorphic malware families. However, due to the effort required to construct the Allaple state machine model, and the number and size of the other polymorphic and metamorphic malware families (perhaps a hundred families, each with only a few thousand artifacts, as contrasted with hundreds of thousands of artifacts for the Allaple family), it is not feasible to construct a state machine model for each family by hand, and automation of this task is clearly required.

The problem of reconstructing a model from samples of its output is called inductive inference. There are many approaches to inductive inference, each with its own application area. A promising approach to the polymorphic and metamorphic malware problem can be found in the Hidden Markov Model (HMM) induction framework of Andreas Stolcke and Stephen Omohundro [2,3,4], originally developed for speech recognition. This work does not directly provide a specific algorithm for automatically creating state machine detectors for polymorphic and metamorphic malware. Rather, it defines solid mathematical foundations and a generalized framework for understanding tradeoffs in the many decisions required to proceed from the Stolcke and Omohundro framework to a specific mathematical and then algorithmic solution.

The objective of this research project is to adapt, extend, and apply the HMM induction framework of Stolcke and Omohundro to address the problem of automatically constructing state machine models for polymorphic and metamorphic malware. This work involves four phases: (1) validating their research findings; (2) specializing their framework to address the particular problem domain of polymorphic and metamorphic malware over the Intel instruction set; (3) extending their algorithms to handle the unique challenges present in this problem domain; and (4) implementing tools based on the research results from the previous phases and deploying them over the entire CERT artifact catalog, thereby reaping the benefits of substantially improved malware identification and classification.

The success of this research will be based on the number of polymorphic and metamorphic malware artifacts that are correctly identified by the HMM-based detectors produced during the final phase of the project. Another important factor will be the extent to which construction of HMM-based detectors can be made automatic, requiring little or no human intervention or supervision. Due to both the complicated nature of the Allaple polymorphic malware family and the large amount of data from the many Allaple instances, automatic construction of the Allaple state machine model would represent a significant research breakthrough.

References

[1] Cohen, C. and Havrilla, J., "Malware Clustering Based on Entry Points," *2008 CERT Research Annual Report,* Software Engineering Institute, Carnegie Mellon University Pittsburgh, PA..

[2] Stolcke, A. and Omohundro, S., "Hidden Markov Model Induction by Bayesian Model Merging," Advances *in Neural Information Processing Systems 5,* eds. S. J. Hanson, J. D. Cowan, and C. L. Giles, pp. 11-18, Morgan Kaufmann, 1993.

[3] *Best-first Model Merging for Hidden Markov Model Induction,* Technical Report TR-94-003, International Computer Science Institute, Berkeley, CA, revised April 1994.

[4] Stolcke, A. and Omohundro, *Model Merging for Hidden Markov Model Induction*, NEC Research Institute, Inc. Technical Report, 1996.

Baselining Port-Specific Scanning Behavior

Rhiannon Weaver

The incident of a new large-scale exploit or worm targeting a particular port can be preceded by a gradual increase in scan activity on that port. But there has been little study in typical behavior of scanners that does *not* lead to an outbreak. How long-lived are scans? How much variation is there in the number of IP addresses scanning a particular port, and in the number of hosts each address scans? In the new era of botnets and proxy networks, it is not immediately clear that a large increase in scanners represents a universal growth in interest on that port. In this research, we wish to examine port-specific scanning activity in order to illuminate some trends about how and why activity changes. This information can be used to help refine data collection and thresholds for port-specific scan activity, with the goal of reducing false positive alarms.

We used the combined Thresholded Random Walk and MISSILE algorithms, built into the SiLK rwscan tool, to flag scanners. In six-hour intervals over a period of November 1 through December 31, 2008, we recorded the number of unique source IP addresses scanning a large network, as well as a summary of the number of unique destination addresses scanned per source IP address. These time series were recorded for each of the 65536 unique ports. As a first step to find surging ports, we performed linear regressions of unique source IP (log-scaled) over time, and ranked ports by the steepness (either increasing or decreasing) of the resulting slope estimate. Figure 1 shows some examples of highly ranked ports, plotted on a log scale.

In the case exemplified by port 16340, the volume of unique scanners increases gradually over time, as hypothesized would occur with a vulnerability announcement. However in other cases, such as ports 8443 and 7193, the number of scanning IP addresses increases rapidly by orders of magnitude. Rather than a continued increase, the high-volume activity stabilizes, often over a period of weeks or months. Rapid declines, as exemplified by port 5579, are also evident. We hypothesize that this activity pattern is due to botnets. The colors of the points represent the number of addresses scanned by the majority of source IP addresses, with green indicating few addresses scanned per source, and redder colors indicating higher numbers. Only in the case of port 8443, during the week of November 17th, does it appear that incoming sources also scan large parts of the internal network.

This kind of botnet activity can lead to false positives in an alerting system looking for new port-based vulnerabilities; none of these incidents were associated with an outbreak. We hypothesize that scanning would not only increase, but that scanners would become bolder in scope, targeting larger areas for scanning of the vulnerable port, as opposed to targeted surgical attacks. In future research we plan to study the use of both unique scanning sources and the number of destination addresses scanned, in order to help pinpoint port-based vulnerabilities and to predict outbreaks more accurately.

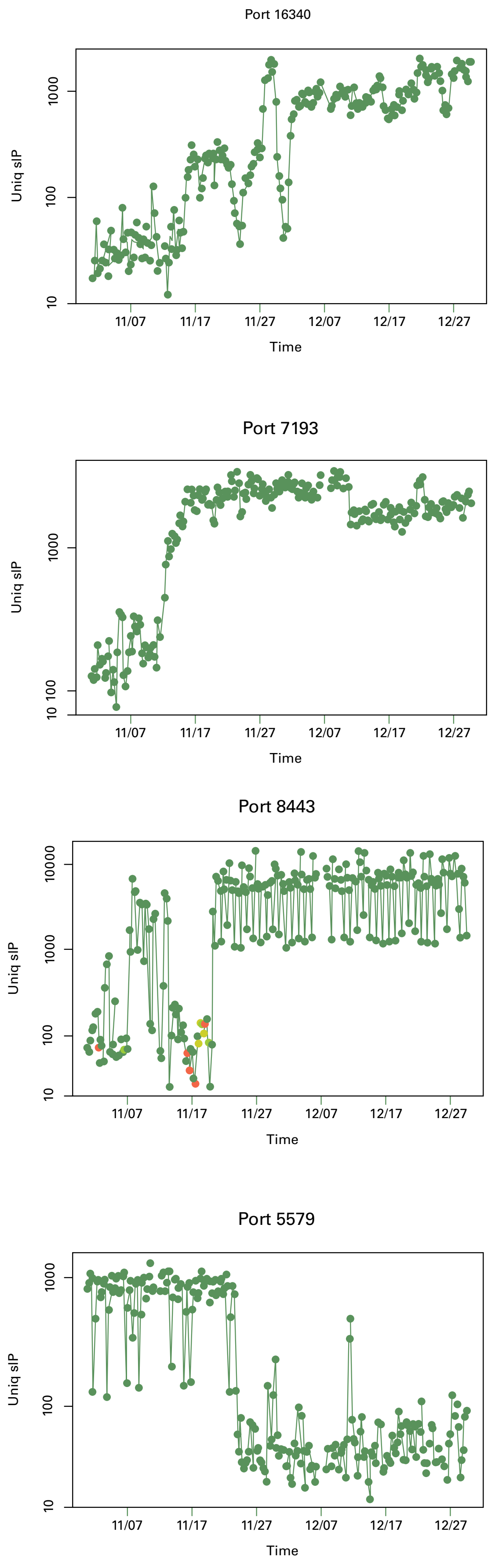


Figure 1: Volume of unique sources scanning several different ports

Building Assured Systems Framework (BASF)

Nancy R. Mead and Julia Allen

Problem Statement

There is no single, recognized framework to organize research and practice areas focused on building assured systems (BAS). If we had such a framework, sponsors could use it to show how research and development in the BAS area fits into the big picture. It could also demonstrate how CERT research efforts fit into the BASF.

Background on Assured Systems

The following items from CERT's internal research planning all in some way address the problem of BAS. Note that these items exhibit varying levels of maturity, and use differing terminology, but they all support this topic.

Engineering Resilient Systems – encompasses secure software engineering as well as requirements engineering, architecture, and design of secure systems and large systems of systems.

Containment – focuses on the problem of how to monitor and detect a component's behavior to contain and isolate the effect of aberrant behavior while still being able to recover from a false assumption of bad behavior.

Architecting Secure Systems – defines the necessary and appropriate design artifacts, quality attributes, and appropriate tradeoff considerations that describe how security properties are positioned, how they relate to the overall system/IT architecture, and how security quality attributes are measured.

Secure Software Engineering (secure coding, software engineering, and hardware design improvement) – improves the way software and hardware are developed by reducing vulnerabilities from software and hardware flaws. This work includes technology life-cycle assurance mechanisms, advanced engineering disciplines, standards and certification regimes, and best practices. The research areas it focuses on include refining current assurance mechanisms and developing new ones where necessary, developing certification regimes, and exploring policy and incentive options.

Secure software engineering encompasses a range of activities targeting security. *Software Security Engineering: A Guide for Project Managers* [1] presents a valuable discussion of these topics. In varying levels of detail, the book examines the spectrum of the appropriate activities:

- requirements Engineering for Secure Software
- secure Architecture and Design
- secure Coding and Testing
- security and Complexity: System Assembly Challenges
- governance and Managing for More Secure Software

However, the discussion of these topics in *Software Security Engineering: A Guide for Project Managers* does not preclude additional research. In fact, several are the focus of CERT's current research projects: security requirements engineering, secure coding, governance and management, and systems complexity.

Current State of the Practice

Some organizations have begun to pay more attention to BAS, including Microsoft SDL [2], other SAFECode consortium members [3], and Oracle. These efforts tend to be stronger in vendor organizations. However, they are weaker in large organizations developing systems for use in-house and integrating across multiple vendors. They are also weaker in small- to medium-size companies developing products for licensed use. Furthermore, there are a variety of life-cycle models in practice—no single approach has emerged as standard. Even in the larger organizations adopting secure software engineering practices, there is a tendency to select a subset of the total set of recommended or applicable practices. Such uneven adoption of BAS suggests the need for ways to measure results.

Approach

In the literature, we typically see life-cycle models or approaches that serve as umbrellas, some of which have been discussed in the book *Software Security Engineering: A Guide for Project Managers* as well as in the article "Software [In] security: A Software Security Framework: Working Towards a Realistic Maturity Model" by Gary McGraw and Brian Chess[4].

The research has started with a review of existing frameworks and lifecycle models for BAS. This activity will be followed by development of a framework that would serve as an umbrella for research into how to build assured systems. We anticipate that this framework would initially be reviewed internally and later on externally. It does not need to be elaborate or lengthy—simplicity is highly desirable.

The expected output is a report describing the BASF and showing the placement of current related research activities within CERT, the SEI, and elsewhere. This report could be used in our research planning, our communications with others, and also in the CERT Research Annual Report. We expect that the DoD and other sponsors would find it useful for tracking current R&D efforts in BAS, and possibly in acquiring assured systems.

References

[1] Allen, J., Barnum, S., Ellison, R., McGraw, G., and Mead, N. R. *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley Professional, 2008 (ISBN-13: 978-0-321-50917-8).

[2] Lipner, S. and Howard M. "The Trustworthy Computing Security Development Lifecycle." March 2005. http://msdn.microsoft.com/en-us/library/ms995349.aspx

[3] http://www.safecode.org

[4] McGraw, C. and Chess, B. "Software [In]security: A Software Security Framework: Working Towards a Realistic Maturity Model." *InformIT*. October 15, 200. http://www.informit.com/articles/article.aspx?p=1271382 .

Control System Security and Critical Infrastructure Survivability

Howard F. Lipson

The complex control systems and devices that automate operation of society's critical infrastructures, including the electric power grid, transportation, oil and natural gas, chemical plants, and manufacturing and industrial processes, are increasingly Internet-connected. Clearly, there are significant economic benefits and efficiency gains for both vendors and utilities for using common Internet-based technologies, built on open Ethernet networking protocols, as the foundation for constructing control system networks and devices. These economic benefits accrue even if such control devices are used solely on private local area networks. Moreover, these open designs allow for the mass production of commercial-off-the-shelf (COTS) control system products that can be used throughout and across industry segments. While this high degree of open-standards-based connectivity brings benefits, it is accompanied by growing risks of targeted cyber attacks that could result in economic and societal consequences of substantial proportions. These attacks are literally enabled by common software vulnerabilities and the inherent susceptibility of networked systems to malicious access.

For example, the Internet and its enabling technologies are playing an increasing role in electric power systems, improving the efficiency and sophistication of business and technical operations. However, Internet connectivity also introduces significant new risks to the power grid's automated control systems and hence to the power grid itself. Moreover, the scientific and engineering foundations for secure and survivable system design must be substantially extended to address the scope and complexity of the sophisticated forms of control envisioned for next-generation energy systems. Equally important is the need for technology guidelines to ensure that engineering best practices are consistently employed during system design, acquisition, development, integration, and evolution.

By undertaking the development, procurement, and deployment of Advanced Metering Infrastructure (AMI) devices and communications networks, the electricity industry is taking the earliest evolutionary steps necessary for realizing the ultimate vision and benefits of smart grid technology. As currently envisioned, smart grid services promise unprecedented levels of automation, situational awareness, efficiency, and fine-grained control of the generation, transmission, delivery, and use of electric power. During FY 2009, members of the Software Engineering Institute from CERT and the Research, Technology, and System Solutions (RTSS) program contributed expert advice and ongoing support to the Advanced Metering Infrastructure Security (AMI-SEC) Task Force [1] of the Smart Grid Security (SG Security) Working Group [2], organized under the Open Smart Grid Subcommittee of the UCA International Users Group. The AMI-SEC Task Force "is charged with developing security guidelines, recommendations, and best practices for AMI system elements" for the benefit of end-users (i.e., utilities) and vendors of AMI technology. In particular, SEI staff participated in the task force's AMI Security Acceleration Project (ASAP) and helped to co-author an *AMI System Security Requirements* document [3], which was ratified by the utility members of the AMI-SEC Task Force and was a "first-of-its-kind" for the utility industry.

SEI staff members continued to make extensive contributions to a new project under the SG Security Working Group, called the Advanced Security Acceleration Project for the Smart Grid (ASAP-SG). ASAP-SG encompasses not only further elaboration and improvement of recommended security controls for developing and procuring AMI systems and devices, but is also beginning to extend security guidance into other emerging areas of smart grid functionality. As part of the ASAP-SG team, SEI staff members collaborated with other members from EnerNex, Southern California Edison, Consumers Energy, InGuardians, and Oak Ridge National Laboratory to co-author key deliverables for the SG Security Working Group. The ASAP-SG team has drafted a *Security Profile for Advanced Metering Infrastructure* document [4] which contains recommended security controls that have been tailored for use with AMI systems. Also under development by the ASAP-SG team is a *Security Profile Blueprint* which will assist domain experts in creating or customizing "security profiles" for major classes of smart grid business functions. An overview document that describes the strategies and guiding principles used in the creation of security guidelines is being drafted, with SEI staff members as the primary authors.

The ASAP-SG team, including SEI staff, also collaborated with and contributed to the efforts of the National Institute of Standards and Technology (NIST) Cyber Security Coordination Task Group (CSCTG) [5]. The NIST CSCTG was formed to "address the cyber security aspects of the Smart Grid Interoperability Framework." Much of the content of the *Security Profile for Advanced Metering Infrastructure* drafted by the ASAP-SG team was incorporated by the CSCTG into a draft NIST interagency report, *Smart Grid Cyber Security Strategy and Requirements* [6], released in September 2009.

In other related activities, CERT contributed cyber security requirements language as part of the merit review criteria for the Department of Energy's Smart Grid Investment Grants Program (under the American Recovery and Reinvestment Act). As has been the case since 2006, CERT has continued to extend its efforts in vulnerability analysis and remediation to include control system networks and devices. A member of CERT also serves as an adjunct research faculty member at the Carnegie Mellon Electricity Industry Center.

CERT plans to continue to expand its operational and research focus on control system security engineering and critical infrastructure survivability in 2010. A primary goal is to use lessons learned from incident and vulnerability analysis to contribute to the creation and enhancement of software engineering methods and best practices for secure and survivable control system development, operation, and assurance. Another major objective is to continue to raise vendor and utility awareness of engineering best practices for control system security and survivability, which includes plans for further SEI contributions to the ASAP-SG team, the SG Security Working Group, and the NIST smart grid cyber security efforts throughout 2010.

References

[1] http://osgug.ucaiug.org/utilisec/amisec/default.aspx

[2] The SG Security Working Group is also known as the Utility Security (UtiliSec) Working Group.

[3] AMI System Security Requirements, Version 1.01. Advanced Metering Infrastructure (AMI) Security Acceleration Project (UCAIUG: AMI-SEC-ASAP), December 17, 2008. http://osgug.ucaiug.org/utilisec/amisec/Shared%20Documents/1.%20System%20Security%20Requirements/AMI%20System%20Security%20Requirements%20-%20v1_01%20-%20Final.doc

[4] Security Profile for Advanced Metering Infrastructure, Draft Version 0.46. Prepared for the UtiliSec Working Group (UCAIug) and the NIST Cyber Security Coordination Task Group by the Advanced Security Acceleration Project for the Smart Grid (ASAP-SG), September 18, 2009. http://osgug.ucaiug.org/utilisec/amisec/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2futilisec%2famisec%2fShared%20Documents%2fAMI%20Security%20Profile%20%28ASAP-SG%29&FolderCTID=&View={7B63C81F-617F-4FC1-AFCB-8404B6B6B0A7}

[5] The NIST CSCTG is transitioning to the Smart Grid Interoperability Panel Cyber Security Working Group (SGIP-CSWG). http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/CyberSecurityCTG

[6] Smart Grid Cyber Security Strategy and Requirements, Draft NIST Interagency Report 7628, The Cyber Security Coordination Task Group (Annabelle Lee, Lead; Tanya Brewer, Editor) and Advanced Security Acceleration Project - Smart Grid, September 2009. http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/NISTIR7628Sep252009

Cyber Assurance

Christopher Alberts, Robert J. Ellison, and Carol Woody

Project Description

Cyber assurance is the justified confidence that networked systems are adequately secure to meet operational needs, even in the presence of attacks, failures, accidents, and unexpected events. This requires appropriate consideration of operational security across all aspects of acquisition, development and deployment, and operations and sustainment.

Existing assurance approaches are primarily single system, single organization focused. With the highly interconnected, complex environments in use today, effective cyber assurance must be addressed across multi-program acquisitions, through the supply chains, and among operational environments that span multiple organizations.

In addition, security considerations are typically handled by experts operating outside of the normal acquisition and development workflow addressing certification and accreditation activities. Instead, cyber assurance must be effectively fused with day-to-day acquisition, development, and operational activities and not viewed as separate add-on actions.

In order to assure the operational security characteristics of networked systems, appropriate methods and metrics for managing and monitoring are critical.

Decisions impacting security are made at multiple levels of the organization as well as across the acquisition life cycle, but there is not an effective means of bridging among the range of stakeholders, which can include program management, architects, system and software engineers, implementation support, security specialists, operational management, and operational support. An integrated decision-making framework is needed that can link a management perspective with the detailed technical and operational realities so that the impact of decisions made at each level can be determined and appropriately evaluated. For example, this integrated view would allow cost and schedule options to be evaluated against the operational security risk.

Research tasks will include

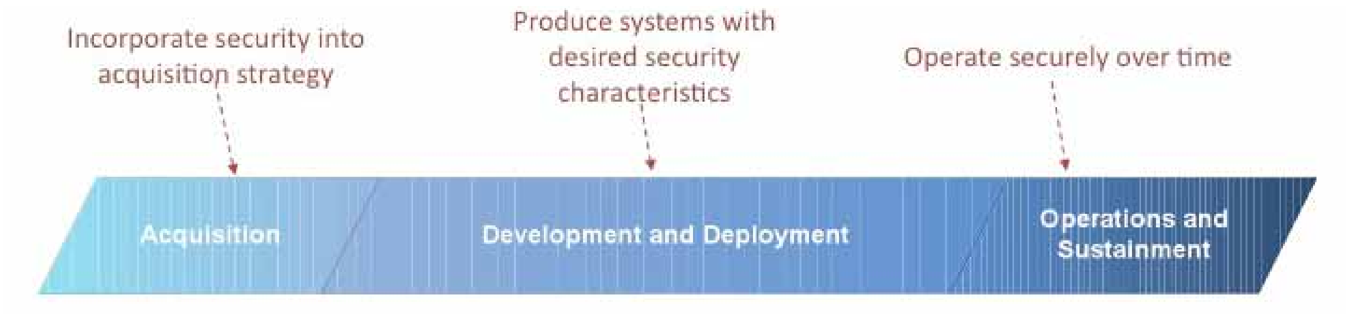
- development of assessment techniques for cyber assurance in multi-system, multi-enterprise environments
- establishing an integrated decision-making framework
- building best practices for cyber assurance relative to acquisition, development and deployment, and operations and sustainment
- identification and use of metrics to monitor and manage cyber assurance
- approaches for using modeling and simulation to analyze and improve cyber assurance

References

Ellison, R., Goodenough, J., Weinstock, C., and Woody, C. Survivability Assurance for Systems of Systems, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2008-TR-008, http://www.sei.cmu.edu/publications/documents/08.reports/08tr008.html

Alberts, C., Smith II, J., and Woody, C. Multi-view Decision Making (MVDM) Workshop, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2008-SR-035, http://www.sei.cmu.edu/publications/documents/08.reports/08sr035.html

Alberts, C., Woody, C., "Consider Operational Security Risk During System Development" published in IEEE Security & Privacy January/February 2007


Incorporate security into acquisition strategy
Produce systems with desired security characteristics
Operate securely over time
Acquisition
Development and Deployment
Operations and Sustainment

Cyber Security Risk Assessment in the Bulk Electric System

Samuel A. Merrell and James F. Stevens

The ability of the United States' electricity infrastructure to deliver power reliably when and where it is needed is a significant component of both of its continued prosperity and national security. The electricity infrastructure in the U.S. consists of three primary components: generation, transmission, and distribution. The generation and transmission components together make up the Bulk Electric System (BES) [1]. As a highly interconnected and interdependent system, the BES requires an approach to risk assessment that accounts for the reliability and operability requirements of individual organizations and of the system as a whole. Unless a risk assessment is able to identify and consistently value shared risks, it is not possible to ensure that organizations are taking actions that are in the best interests of grid reliability.

To address this need, the CERT Program developed the OCTAVE BES method, a cyber security risk assessment approach specifically tailored for the functions that ensure the safe and reliable generation and transmission of electricity. CERT has a rich history in developing successful structured risk assessments for the purposes of cyber security management. These include the OCTAVE Method, OCTAVE-S, and most recently OCTAVE Allegro. These assessments serve specific audiences but share a fundamental structural foundation. All of them have been effective in assisting organizations in identifying their cyber security risks and in supporting the development of risk mitigation plans.

OCTAVE BES leverages the fundamental approach to cyber risk assessment that has proven to be successful with the other OCTAVE methods, but provides a tailored methodology to suit the needs of the organizations that comprise the sector and focuses on BES reliability and operability. OCTAVE BES defines a process to support the identification and assessment of cyber risks to the BES. It facilitates an organization's ability to internalize resiliency requirements of the BES, and encourages uniform risk assessment and risk management across the BES.

There are eight steps in the OCTAVE BES method as shown in Figure 1. The process begins with the selection and profiling of a BES function to assess and concludes with a collection of risk statements. These risk statements are a view of the current risk environment for a given BES function.

OCTAVE BES is being developed in conjunction with owners and operators of BES assets. Working directly with sector members to develop OCTAVE BES will provide a number of benefits for the assessment methodology that will allow it to be more likely to be

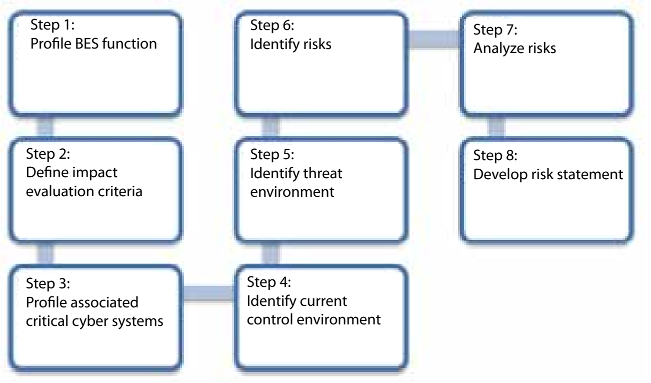

Step 1:
Profile BES function
Step 2:
Define impact evaluation criteria
Step 3:
Profile associated critical cyber systems
Step 4:
Identify current control environment
Step 5:
Identify threat environment
Step 6:
Identify risks
Step 7:
Analyze risks
Step 8:
Develop risk statement

Figure 1: An overview of OCTAVE BES

- readily applicable to the BES
- applied correctly
- accepted by the sector members that will be using it

In addition to working directly with the owners and operators of BES assets, efforts were made to align OCTAVE BES with current and emerging NERC CIP standards and guidance wherever possible. This will, hopefully, ensure that the method will support compliance efforts while also facilitating the ability of BES organizations to manage their cyber security risks effectively and efficiently.

An initial pilot of the OCTAVE BES method was conducted in August 2009. The piloting process began with simply identifying a candidate BES function to assess. The function of Electricity Transmission Control was selected as it is conceptually and operationally straightforward—a master SCADA system gathers information and issues commands to relays in the field. Selection of a straightforward function was purposeful for this initial piloting activity because it allowed staff from both the SEI and the pilot organization to focus on the assessment process and not on explaining and understanding details about the mechanics of a more complex BES reliability function.

The initial piloting activities have demonstrated the effectiveness and usability of the OCTAVE BES, but additional research and development is required. Two of the highest priority tasks for this coming year are to continue to refine the guidance and supporting materials so that individual organizations can conduct assessments without expert facilitation and to perform additional pilots on more complex BES functions and organizations to determine if the method scales and to improve its overall performance.

References

[1] NERC defines the Bulk Electric System as generation resources, transmission lines, interconnections with neighboring systems, and associated equipment, generally operated at voltages of 100 kV or higher.

Influencing National Capability Development in Cyber Security through Incentives

Bradford Willke and Samuel A. Merrell

The development of a national capability for cyber security is lacking throughout the majority of the nations of the world. Of the 194 nations recognized by the U.S. Department of State, less than 90 have a formal, established organization that is capable of responding to a computer security incident, let alone manage the other activities that a national cyber security program must oversee, such as those proposed by the United Nations' International Telecommunications Union.

The United States should harness opportunities to promote the consideration and development of robust cyber security capability in emerging economies. Through the creation of incentive programs that accompany foreign aid and investment, the U.S. can influence foreign protection of information infrastructure and create a capacity for performing national cyber security. When the U.S. provides funding to aid organizations such as the International Monetary Fund (IMF), the World Bank, or the International Red Cross, it can seize the opportunity to encourage desirable behaviors, such as the adoption of cyber security.

As an example, the IMF currently administers a $602 million loan for the African nation of Ghana. This loan was granted to assist Ghana in the development of infrastructure to support oil production. The information technologies that support such an infrastructure, such as SCADA systems and corporate computing environments, are highly important to resilient oil production. If these assets are afforded an inadequate level of cyber security, including ineffective management of cyber threats, vulnerabilities, consequences, and risk, they could compromise the integrity of the very infrastructure that they will be designed to control and support.

The United States has an important opportunity to encourage global safety and stability through incentive programs that accompany foreign aid. U.S. and international news headlines are awash with accounts of malevolent actors infiltrating critical information and service delivery infrastructures across the globe. It is in the best interest of the U.S. to compel both the discourse to raise cyber security awareness and the means by which cyber security is practiced. It is also incumbent on the U.S. to lead emerging economies to improvements in cyber security to ensure that these situations can be adequately prepared for and prevented. Developing a national capability for cyber security management contributes to overall national security and stability.

Measuring Operational Resiliency

Julia Allen

Since 2003, CERT has conducted research to develop a model for managing operational resiliency. CERT researchers drew from the SEI's experiences with CMMI and the CERT OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) method, collaboration with leaders in the U.S. financial services sector, and existing codes of practice in security, business continuity, and IT operations. The resulting CERT Resiliency Management Model (RMM) [1] describes the processes that characterize the organizational capabilities necessary to manage operational resiliency. In FY09, collaboration partners and customers assessed their organizational performance against the model to benchmark current practice, validate the model, and begin process improvement efforts.

Also in FY09, CERT commenced a new area of research to identify candidate measures for determining the effectiveness of the processes described in the CERT RMM in improving operational resiliency. The first step in defining a meaningful measurement program is to determine the required or desired level of operational resiliency for an organization. ("Organization" may be the enterprise, any business line or operating unit of the enterprise, or a supply chain or other form of business relationship that includes external entities.) The CERT RMM provides a process-based structure of goals and practices at four levels of maturity (Incomplete, Performed, Managed, and Defined) and a companion appraisal method. Defining the required or desired level establishes the baseline against which operational resiliency can be measured. Ideally, the required level is established during strategic and operational planning as well as when planning for continuity of operations, not as an afterthought during times of stress and service disruption. The required level should be no less than, and no more than, that which is required to meet business mission objectives. The purpose of CERT's new area of research is to answer the question, "What should be measured to determine whether the required level of operational resiliency has been achieved?"

Example measures for answering this question for each process area appear in CERT RMM version 1.0. They include key process measures and performance indicators that can be used, in whole or in part, to demonstrate that the required level of operational resiliency has been achieved for a given process area. Examples of measures for 5 of the 26 process areas are included in Table 1.

Table 1: Examples of Measures in Selected RMM Process Areas

Process Area	Example Measures
Knowledge and Information Management	• percentage of information assets that do not have stated owners or custodians • frequency and timeliness of information asset backups; frequency of backup restoration testing • percentage of information assets for which encryption is required and not yet implemented
Technology Management	• percentage of technology assets (software, hardware, systems) for which the cost of compromise (loss, damage, disclosure, disruption in access to) has been quantified • percentage of technology assets for which some form of risk assessment has been performed as required by policy • number of unauthorized changes to technology assets during a stated time interval
Service Continuity	• percentage of service continuity plans tested (and number of times tested by time period) • percentage of service continuity plans that failed one or more test objectives • percentage of high-value services and supporting assets that do not have service continuity plans • percentage of unmet recovery time objectives and recovery point objectives
Vulnerability Analysis and Resolution	• percentage of high-value assets that have been monitored, assessed, and audited for vulnerabilities within a stated time interval • percentage of vulnerabilities that have been satisfactorily remediated, by time interval
Resiliency Requirements Definition	• percentage of services and assets for which resiliency requirements have been defined and documented (or conversely, for which requirements are not stated or are incomplete) • elapsed time between identification of new assets and the development of resiliency requirements for those assets

Research tasks in FY10 are focused on the identification, deployment, piloting, and measurement of effective security and resiliency measures. These measures may be performance or process based. Automated approaches for collecting and reporting measures will be considered. Tasks include reviewing and extending the process area measures and measurement and analysis description that appear in version 1.0, working with collaborators and customers to determine what measures are most useful for determining process effectiveness, developing measures templates and structured definitions, and updating the model to reflect results.

This research also includes an examination of how statistical process control concepts can be used in resiliency and security measures and in determining how much resiliency is sufficient. This may involve analyzing various combinations of CERT RMM process areas and process area measures to help decision makers better specify what levels of operational resiliency are required and desired, and measures for determining when and if those levels have been achieved.

References

[1] CERT Resilient Enterprise Management Team. *CERT Resiliency Management Model, v1.0*. Software Engineering Institute, Carnegie Mellon University, 2009. http://www.cert.org/resiliency/rmm.html.

Measuring Software Security

Julia Allen

Since the mid 1990s, CERT has researched and created value-added processes, methods, practices, and tools for software survivability, software assurance, and building security into software throughout its development life cycle. In recent years, the research community has increasingly contributed to the body of knowledge about software assurance and software security metrics.

Unfortunately, the security community often conflates information security metrics and software security metrics, which in fact are quite distinct. Efforts to identify meaningful information and operational security metrics have been ongoing for some time. These efforts include various reports by the U.S. National Institute of Standards and Technology [1], the Workshop on the Economics of Information Security (WEIS) [2], and consensus efforts such as those conducted by the Center for Internet Security [3] and the 2004 Corporate Information Security Working Group [4]. However, while they inform and influence one another, information security metrics are not software security metrics.

Consequently, in FY09, CERT began new research in software security measures that builds on CERT's core competence in software and information security. The purpose of this research is to address the following two questions:

- How do I establish and specify the required/desired level of security for a specific software application, set of applications, software-reliant system, system of systems, supply chains, and other multi-system environments?

- How do I measure, at each phase of the development or acquisition life cycle, that the required and/or desired level of security has been achieved?

Approaches to answering the first question define the baseline against which software security can be measured. Such approaches create a meaningful measure of the degree of software security for a specific set of related software components. Ideally, this measurement is performed as part of initial planning and specification, not as an afterthought during testing and integration.

In addition to demonstrating that security requirements are satisfied, risk analysis approaches, including the prioritization of software components based on their contribution to mission success, are also relevant. The SEI has undertaken promising work to identify methods, such as assurance cases, for capturing this expression [5]. This research task will examine the suitability of these methods in establishing a foundation for measuring software security. It will also recommend a range of alternatives with appropriate selection criteria. Software development project managers and stakeholders will be able to select from these alternatives to define a required level of security as part of their software validation criteria.

Given a baseline against which to measure, approaches to the second question will include key product measures, process measures, and performance indicators that can be used to validate the required level of software security appropriate to a given life cycle phase. Such measures will be developed within the context of a measurement process and framework that can be tailored for a specific development project. Table 1 presents early examples of life-cycle-phase measures that could be used to validate required levels of software security:

Research tasks in FY10 include
- investigating existing bodies of knowledge to lay the foundation for addressing the two presented research questions
- building relationships with key thought leaders and potential collaborators
- identifying core definitions
- developing an initial software security measurement process
- publishing initial findings

Research tasks in FY11 and beyond include
- organizing FY10 results by software development life cycle phase to inform the development of a software security measures framework and updated process
- identifying software security measures for acquisition—defining measures that can be written into requests for proposal (RFPs), contracts, service level agreements, and to assist in making funding decisions
- integrating software security development and acquisition measures into selected security assessment and evaluation instruments as well as selected software development and measurement standards

References

[1] Chew, Elizabeth, et. al. *Performance Measurement Guide for Information Security: Special Publication 800-55 Revision 1*. National Institute of Standards and Technology (NIST), July 2008. http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf. Additional NIST reports are available at http://csrc.nist.gov/publications/PubsSPs.html.

[2] The Ninth Workshop on the Economics of Information Security (WEIS 2010). http://weis2010.econinfosec.org/index.html. Agendas and papers presented from prior WEIS events are available online.

[3] The Center for Internet Security Consensus Information Security Metrics. http://cisecurity.org/securitymetrics.html.

[4] Corporate Information Security Working Group. Adam H. Putnam, Chairman; Subcommittee on Technology, Information Policy, Intergovernmental Relations & the Census Government Reform Committee, U.S. House of Representatives. "Report of the Best Practices and Metrics Teams." November 17, 2004; updated January 10, 2005. http://net.educause.edu/ir/library/pdf/CSD3661.pdf.

[5] Work in assurance cases at the Software Engineering Institute: http://www.sei.cmu.edu/dependability/tools/assurancecase/index.cfm.

Life cycle phase	Example software security measures
Requirements engineering	• Percentage of relevant software security principles reflected in requirements specifications (this assumes that security principles essential for a given development project have been selected) • Percentage of security requirements that have been subject to analyses (risk, feasibility, cost/benefit, performance tradeoffs) prior to being included in the specification • Percentage of security requirements covered by attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis
Architecture and design	• Percentage of architectural/design components subject to attack surface analysis and measurement • Percentage of architectural/design components subject to architectural risk analysis • Percentage of high-value security controls covered by security design patterns
Coding	• Percentage of software components subject to static and dynamic code analysis against known vulnerabilities and weaknesses • Percentage of defects discovered during coding that was injected in architecture and design; in requirements specification • Percentage of software components subject to code integrity and handling procedures, such as chain of custody verification, anti-tampering, and code signing
Testing	• Percentage of defects discovered during testing that was injected in coding; in architecture and design; in requirements specification • Percentage of software components with demonstrated satisfaction of security requirements as represented by a range of testing approaches (functional, risk-based, fuzz, penetration, black box, white box, code coverage, etc.) • Percentage of software components that demonstrated required levels of attack resistance and resilience when subject to attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis

Table 1: Example Software Security Measures by Life Cycle Phase

SiLK: Improvements and Plans

Mark Thomas and Michael Duggan

The System for internet Level Knowledge (SiLK) supports the collection and analysis of network flow data. The release of SiLK 2 improves the types of analysis that can be performed.

An analyst frequently wants to group flow records based on selected attributes of the records and compute a value for all records in the group. SiLK uses the term *aggregate value* to refer to a value computed across multiple records. For example, an analyst may ask how many bytes of data were exchanged between each of the hosts inside her network with hosts outside her network. By sorting the byte counts from largest to smallest, an analyst can create a list of the "top-N talkers." Recent improvements to the SiLK tool set allow creating a top-N list for any set of attributes, and greatly increases the types of aggregate values that can be computed. The rwuniq tool has had long-standing support for grouping records using an arbitrary, user-defined key based on attributes of the flow records. In addition, rwuniq allowed an analyst to define new key fields by writing C or Python code ("plug-ins") that could be loaded into rwuniq. rwuniq supports computing multiple aggregate values for each group of records (e.g., sum of packets and sum of bytes), but prior to SiLK 2 the list of aggregate values was fixed to those built into rwuniq. Realizing that it is impossible to predict all the possible aggregate values an analyst may want to compute, SiLK 2 allows an analyst to write C or Python plug-ins that define new aggregate values. To make defining new aggregate values easy, the Python interface provides simple interfaces for defining aggregate values that compute sums, maximums across sets of values, and minimums across sets of values.

The rwstats tool computes top-N (or "bottom-N") lists. In SiLK 1, rwstats had a very limited set of key fields it supported, and it only supported computing a single aggregate value for each group. For data with a large number of unique keys, rwstats would sometimes exit before completion because it had exhausted the machine's memory. To work around the limitations of rwstats, an analyst would use rwuniq to group the records and compute the aggregate values, and then use UNIX utilities to sort the textual output from rwuniq to get the top-N list. While this worked, it was inefficient. The rwstats tool in SiLK 2 is greatly improved: it now supports an arbitrary user-defined key; it supports multiple aggregate value fields; it allows the user to load C or Python plug-ins that define new key fields or aggregate value fields; and it can store data in temporary files when its memory requirements exceed those of the system.

SiLK 2 includes improvements to PySiLK, an extension module for the Python programming language that makes it easy to work with SiLK flow records from Python. PySiLK now supports manipulating bag and prefix map files, two binary SiLK file formats that provide volume and labeling information for IP addresses. When writing Python plug-ins for SiLK tools, creating new key fields is now easier thanks to simplified field registration functions, and analysts can now add new command line options from their Python plug-ins.

Prefix map files are binary files that map an IP address to a textual label. These files can be used for filtering, sorting, and grouping flow records. Prior to SiLK 2, the tools only allowed a single prefix map to be used per invocation. The tools now support multiple prefix maps, and additional SiLK tools now have prefix map support.

Additional changes were made across the SiLK tool set to improve IPv6 support and to make the tools more functional and more consistent.

CERT is planning continued improvements to SiLK in 2010, but the primary focus will be in making it easier to run routine analysis on incoming flow data and to allow querying of SiLK flow data using the SQL language, which will allow SiLK to interoperate with other sources of data more easily.

The Smart Grid Maturity Model

James F. Stevens and David W. White

The Smart Grid Maturity Model (SGMM) is a management tool that provides a framework for utilities to plan, measure progress, and prioritize options as they move towards the implementation of a smart grid. The SGMM focuses on utilities that have a strong distribution component to their operations, but is designed to meet the needs of a wide range of electric utilities: public or investor owned, rural or urban, large or small, foreign or domestic, and those that have generation capacity or rely on power purchase agreements to meet their customer load. A very wide range of demographics were considered in developing the model so that it would be broadly applicable.

The SGMM was developed in 2007 by IBM and the Global Intelligent Utility Network Coalition with the assistance of APQC. After the initial development was completed it was determined that the model should be transferred to a global body or institution for totally impartial delivery, stewardship, and growth. The Software Engineering Institute (SEI) was selected to become the steward of the SGMM and officially took over that role in March 2009. As a global leader in software and systems engineering and architecture, security best practices, process improvement, and maturity modeling, the SEI is uniquely positioned to take on the challenges leading the future development of this model.

The model itself consists of eight domains of related capabilities and characteristics against which a utility can assess its current state and its desired future state. The domains, which provide focus areas for evaluating capability, are as follows:

1. Strategy, Management, and Regulatory
2. Organization and Structure
3. Grid Operations
4. Work and Asset Management
5. Customer Management and Experience
6. Technology
7. Value Chain Integration
8. Societal and Environmental

Achieving a balance in performance across these domains within an organization demonstrates that the organization has examined and improved upon a broad portion of its operations.

A six-point scale (0-5) is used to indicate the maturity for each domain in the model. Level 0 is the default and simply indicates that an organization has not achieved the necessary capabilities and characteristics required to achieve Level 1 in a given domain. Level 5 indicates that an organization has fully achieved the capabilities and requirements and is perhaps moving beyond simple smart grid implementation.

To develop its maturity profile, a utility completes an assessment survey, which can be performed by the utility itself or through a facilitated workshop. An impartial third party then scores the completed survey. The SEI has partnered with APQC [1] to provide scoring services for the assessment survey.

In September of 2009 the SEI issued the first annual report on the model results titled "*SGMM: First Annual Report on Smart Grid Implementation*." [2]. It is envisioned that this report will be updated each year and released sometime in the late third or early fourth quarter. The 2009 report provided an overview of the results from the 53 utilities that had completed survey assessments by July 2009. Figure 1 below provides a summary of the domain scores. Approximately 53% of the utilities participating in the survey were based in the United States; the rest of the participants operated in other regions of the world. As expected, the results indicated that most utilities were just beginning their smart grid journeys.

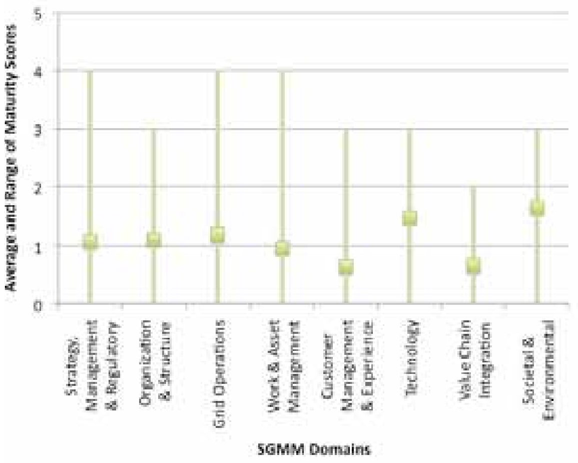


Figure 1: Average and range of maturity scores by domain

With the support of the Department of Energy and the National Energy Technology Laboratory, the SEI and its collaborator APQC are maintaining and evolving the Smart Grid Maturity Model as a resource for industry transformation. In 2010 the SEI has a very aggressive plan to improve the model, to expand the suite of products available to facilitate its use, and to transition its adoption in the community. In the first quarter of 2010 the SEI will release an incremental update to the model focused on addressing structural issues and improving its usability. Shortly after the model update is released, additional training on the model and its facilitated use within a utility will be available. Finally, an aggressive push to increase the number of utilities that have taken the SGMM assessment will be made with the goal of publishing the new results in a 2nd annual report, targeted for late third quarter 2010.

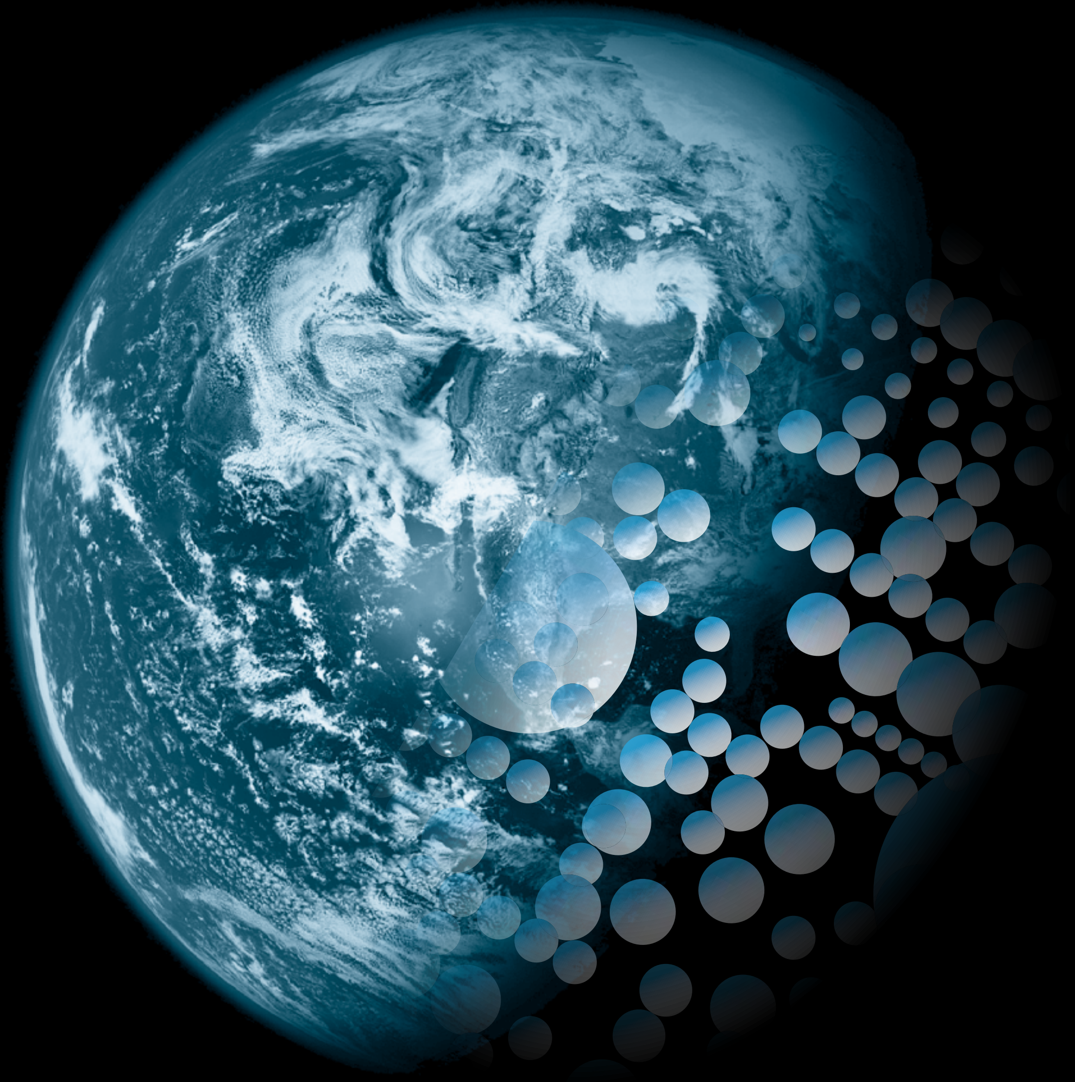
Our hope for this model is that as more organizations participate, the SGMM will become an increasingly valuable benchmarking, best practice, and strategic planning management tool for individual organizations and for the utility industry as a whole. Additional information on the model and notifications about updates can be found at www.sei.cmu.edu/smartgrid.

References

[1] APQC is a non-profit member-based research organization with more than 30 years of systematic quality and process improvement research experience. APQC is working in collaboration with the SEI to evolve the SGMM and to analyze and maintain the data collected.

[2] The report is available at the following URL: http://www.sei.cmu.edu/library/assets/sgmm.pdf

Researcher Activities



List of Selected Publications

Chris Alberts

Reports

Alberts, C. and Dorofee, A. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/reports/09tr007.pdf

Alberts, C., Smith II, J., and Woody, C. *Multi-View Decision Making (MVDM) Workshop* (CMU/SEI-2008-SR-035). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/reports/08sr035.pdf

Papers

Alberts, C., Moore, A., Siviy, J., and Woody, C., *"Value mapping and Modeling SoS Assurance Technologies and Supply Chain"*, 2009 IEEE International Systems Conference; Vancouver Canada, May 2009. Won best paper award.

Julia Allen

Book Chapters

Allen, Julia; Kim, Gene; Love, Paul; Spafford, George. Chapter 3, "IT Operational Pressures on Information Security," *Enterprise Information Security and Privacy*. Artech House, 2009.

Papers

Siviy, Jeannine; Moore, Andrew; Alberts, Christopher; Woody, Carol; Allen, Julia. "Value Mapping and Modeling SoS Assurance Technologies and Assurance Supply Chain." IEEE Systems Council Conference, March 2009. Received best paper award.

Reports

Wrubel, James; White, David; Allen, Julia. *High Fidelity e-Learning: SEI's Virtual Training Environment* (CMU/SEI-2009-TR-005). Software Engineering Institute, Carnegie Mellon University, January 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tr005.cfm.

Robert J. Ellison

Reports

Ellison, R., Goodenough, J., Weinstock, C., and Woody, C. Survivability Assurance for Systems of Systems, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2008-TR-008, http://www.sei.cmu.edu/publications/documents/08.reports/08tr008.html

Howard Lipson

Book Chapters and Sections

Lipson, H., Goodenough, J., & Weinstock, C. Section 2.4, "How to Assert and Specify Desired Security Properties," 61–70. *Software Security Engineering: A Guide for Project Managers*. Addison-Wesley, 2008.

Reports

Brown, B., Singletary B., Willke B., Bennett, C., Highfill, D., Houseman, D., Cleveland, F., Lipson, H., Ivers, J., Gooding, J., McDonald, J., Greenfield, N., & Li S. *AMI System Security Requirements –Version 1.01*, Advanced Metering Infrastructure (AMI) Security Acceleration Project, AMI Security Task Force, UCA International Users Group, December, 2008, 111 pp.

Papers

Lipson, H. & van Wyk, K. "Application Firewalls and Proxies – Introduction and Concept of Operations." Department of Homeland Security *Build Security In* website, Revised September 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/assembly/30-BSI.html

Lipson, H. "Evolutionary Design of Secure Systems – The First Step Is Recognizing the Need for Change." Department of Homeland Security *Build Security In* website, Revised October 2008. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/assembly/467-BSI.html

Lipson, H. Update of "Source Code Analysis Tools – Overview" by C. Michael & S. R. Lavenhar, "Source Code Analysis Tools – Example Programs" by Cigital, Inc., and "Source Code Analysis Tools – Bibliography" by Cigital, Inc. Department of Homeland Security *Build Security In* website, February 2009. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/tools/code.html

Nancy R. Mead

Conference Papers in Published Proceedings

S. Miyazaki, N. R. Mead, J. Zhan, Computer-Aided Privacy Requirements Elicitation Technique, in APSCC: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference, December 2008, pp.367-372.

Mead, N.R., Drommi, A., Shoemaker, D., Ingalsbe, J., A study of the Impact on Students: Understanding Cross Cultural Differences in Software Engineering Work, COMPSAC (International Computer Software and Applications Conference) July, 2009, Seattle, WA

Abu-Nimeh, S., Miyazaki, S., Mead, N.R., Integrating Privacy Requirements into Security Requirements Engineering, IEEE Software Engineering and Knowledge Engineering (SEKE) Conference, July 2009, Boston, MA

Mead, N.R., Shoemaker, D., Ingalsbe, J., Teaching Security Requirements Engineering Using SQUARE, 4th International Workshop on Requirements Engineering Education and Training (REET), IEEE Requirements Engineering Conference, September, 2009

Abu-Nimeh, S., Mead, N,R., Privacy Risk Assessment in Privacy Requirements Engineering, Second International Workshop on Requirements Engineering and Law (RELAW), IEEE Requirements Engineering Conference, September, 2009

Mead, N.R., Shoemaker, D., Ingalsbe, J., Ensuring Cost Efficient and Secure Software through Student Case Studies in Risk and Requirements Prioritization, HICSS 42, January 2009, Hawaii

Nancy R. Mead, continued

Journal Articles

(Invited) Mead, N.R., Software Engineering Education: How Far We've Come and How Far We Have To Go, Journal of Systems and Software, (2009), doi:10.1016/j.jss2008.12.038

(Invited) Mead, N.R., Yoshioka, N., Square Up Your Security Requirements Engineering with SQUARE, Information Processing Society of Japan (IPSJ) Journal, Vol.50 No.3 Mar. 2009

Mead, N.R., Shoemaker, D., Ingalsbe, J., Software Assurance Practice at Ford: A Case Study, CrossTalk, Vol. 22, No. 3, March 2009, pp. 4-7.

Journal Editorships

REJ (Requirements Engineering Journal), Editorial Board

Editorial Review Board – International Journal on Secure Software Engineering (IJSSE), IGI Global

Books Published

Mead, N.R., Shoemaker, D., Book Chapter Novel Methods of Incorporating Security Requirements Engineering into Software Engineering Courses and Curricula, Chapter VI, Software Engineering: Effective Teaching and Learning Approaches and Practices, Eds, Ellis, Demurjian, & Naveda, IGI Global, pp. 98-113, 2008

SEI Reports

Nancy R. Mead, Julia Allen, W. Arthur Conklin, Antonio Drommi, John Harrison, Jeff Ingalsbe, James Rainey, Dan Shoemaker, *Making the Business Case for Software Assurance,* CMU/SEI-2009-SR-001 Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April, 2009

Varokas Panusuwan, Prashanth Batlagundu, Nancy R. Mead, Faculty Advisor, *Privacy Risk Assessment Case Studies in Support of SQUARE,* CMU/SEI-2009-SR-017 Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July, 2009 www.sei.cmu.edu/publications/documents/09.reports/09sr017.html

Robert Seacord

Robert C. Seacord. The CERT C Secure Coding Standard. Addison-Wesley. October 2008. ISBN-13: 978-0-321-56321-7.

Robert C. Seacord. Secure Coding in C and C++. Addison-Wesley. September 2005. ISBN: 0321335724.

Robert C. Seacord, Daniel Plakosh, Grace A. Lewis, Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices, Addison-Wesley, February 2003.

Kurt C. Wallnau, Scott A. Hissam, Robert C. Seacord, Building Systems from Commercial Components, Addison-Wesley, June 2001, ISBN: 0201700646.

Carol Woody

Publications

Ellison, R., Goodenough, J., Weinstock, C., and Woody, C. Survivability Assurance for Systems of Systems, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2008-TR-008, http://www.sei.cmu.edu/publications/documents/08.reports/08tr008.html

Alberts, C., Smith II, J., and Woody, C. Multi-view Decision Making (MVDM) Workshop, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2008-SR-035, http://www.sei.cmu.edu/publications/documents/08.reports/08sr035.html

Alberts, C., Moore, A., Siviy, J., and Woody, C., "Value mapping and Modeling SoS Assurance Technologies and Supply Chain", 2009 IEEE International Systems Conference; Vancouver Canada, May 2009. Won the Best Paper Award.

Talks/Panels/Workshops

Julia Allen

Talks/Panels/Workshops

Allen, Julia. "Making the Business Case for Software Assurance" SecureIT Conference keynote presentation, Los Angeles, California, March 2009. Also presented at the SEI's SEPG North America conference in March 2009.

Allen, Julia; Payne, Shirley. "Making the Case for Security: A Framework for Communicating and Governing." Half-day seminar, EDUCAUSE Security Professionals Conference, Atlanta, Georgia, April 2009.

Podcasts

Captured 20 podcasts as part of the CERT Podcast Series: Security for Business Leaders. http://www.cert.org/podcasts.

Richard Linger

Invited lectures at University of South Florida, March, 2009
"Software Assurance through Correctness Verification"

"Incremental System Development for Risk and Cost Reduction"

"Function Extraction Technology for Software Behavior Computation"

Howard Lipson

Talks/Panels/Workshops

Lipson, H. "Towards a CERT Coordination Center for Control Systems –The Survivability Challenge," 2008 ACS Control Systems Cyber Security Conference, Chicago, Ill., August 2008.

Lipson, H. "Beyond Cyber Security: Built for Survival," Instinctive Computing International Workshop, Carnegie Mellon University, Pittsburgh, Pa., June 2009.

Nancy R. Mead

Webinars

Invited talk on BSI and Software Security Engineering book, FSTC Roundtable, July13, 2009, "Software Security Engineering: How to Get Started"

Keynote speaker for BrightTalk Threat Management Summit September 8, 2009, " Uses of Threat Modeling in Software Development"

DoD Software Collaborators' Telecon, August 4, 2009 "A Toolbox for Improved Security", co-presented with Carol Woody

SQUARE Overview, May 14, 2009, SEI Webinar

Workshops and Tutorials

2-day workshop on SQUARE for National Defense University of Taiwan, May 2009

Knowledge Transfer Network (KTN) Workshop Panelist in March 2009

Co-presenter of tutorial "Software Security Engineering: How to Get Started" DHS Forum in Fall 2008

Robert Seacord

SD West 2009 Conference & Expo

March 9-13
Santa Clara Convention Center, Santa Clara, CA
http://www.SDExpo.com

Title: Secure Coding in C++: Integers
Date: Thursday, March 12

Title: Secure Coding in C++: Strings
Date: Thursday, March 12

Title: Dangerous Optimizations and the loss of Causality
Date: Friday, March 13

Software Development Best Practices

Tuesday, October 28, 2008
Secure Coding in C and C++: Strings

Secure Coding in C and C++: Integers

Carol Woody

Presentations

MORS Symposium, "Evaluating the Impact of Technology Change on Mission Success – a Process Approach", Ft Leavenworth KA, May 2009

DHS SwA Working Group, "Overcoming Barriers of Adoption for Software Assurance", Laurel MD, December 2008

AIAA, "The Landscape of Software Assurance – Participating Organizations and Technologies", AIAA Unmanned... Unlimited Conference, Seattle WA, April 2009

DHS SwA Forum, "The Way Forward for Mitigating Software Supply Chain Risk", Arlington VA, November 2009

Technical Leadership

Julia Allen

Technical Leadership
CERT representative to the EDUCAUSE Security Task Force Risk Management Working Group. Served as subject matter expert to capture best practices for information security governance (as a wiki site).

Howard Lipson

Co-organizer, Instinctive Computing International Workshop, Carnegie Mellon University, June 2009. http://www.cmu.edu/vis/workshop-brochure-v6.pdf

Invited presentation, "Towards a CERT Coordination Center for Control Systems –The Survivability Challenge," 2008 ACS Control Systems Cyber Security Conference, Chicago, Ill., August 2008

Session chair, "Privacy in Services," Fourth ACM Workshop on Digital Identity Management, ACM CCS, October 2008

Program committee member, Fourth ACM Workshop on Digital Identity Management, ACM Conference on Computer and Communications Security, October 2008

Program committee member, Fifth ACM Workshop on Digital Identity Management, ACM CCS, November 2009

Program committee member, International Workshop on Security Measurements and Metrics (MetriSec), October 2009

Program committee member, Workshop on Ethics in Computer Security Research (WECSR), January 2010

Served in an expert advisory role for the Smart Grid Security Working Group's Advanced Security Acceleration Project for the Smart Grid

Research Fellow, Instinctive Computing Laboratory, Carnegie Mellon University

Member (founding), Carnegie Mellon Electricity Industry Center

Member of the Advisory Board for Duquesne University's Graduate Program in Computational Mathematics, 1999-present (Chair 2002-2004)

Nancy R. Mead

IEEE Fellow, Distinguished Member of the ACM

ISA (Information Security & Assurance) Conference 2009, Program Committee

PASSAT (IEEE International Conference on Information Privacy, Security, Risk and Trust) Conference 2009, Program Committee, August, Vancouver, Canada

GSwERC Invited Reviewer 2009

COMPSAC (International Computer Software and Applications Conference) 2009, Program Committee

Program Committee for the Workshop on Privacy (RELAW) held in conjunction with the Requirements Engineering Conference RE'09

Program Committee for the Workshop on E-Voting held in conjunction with the Requirements Engineering Conference RE'09

Robert Seacord

CMU representative to INCITS PL22 – Programming Languages - U.S. TAG to ISO/IEC JTC1/SC22

CMU representative to INCITS PL22.11 – Programming Language C

Technical Expert ISO/IEC JTC1/SC22/WG14 – Programming Language C

Technical Expert ISO/IEC JTC1/SC22/WG23 – Language Vulnerabilities

PL22 Liaison to CS1 - Cyber Security –US TAG for ISO/IEC JTC 1/SC 27 and all SC 27 Working Groups.

Financial Services Technology Consortium (FSTC) Software Assurance Initiative Secure Coding Work Stream Leader

Carol Woody

Elected IEEE Senior Member, August 2009

Elected ACM Senior Member, August 2009

Session Co-chair for the Military Operations Research Society (MORS) Workshop "Understanding the Consequences of Catastrophic Events: Using Methods and Tools to Analyze and Manage Incidents", November 2008

Panel Facilitator, DHS SwA Forum, "Where is Academia Going and How Can the SwA Forum Help", Arlington VA, November 2009

Biographies

Christopher Alberts

Christopher Alberts is a senior member of the technical staff at the SEI. He has 25 years of experience in software engineering and information technology, with particular emphasis on systems analysis, software design and development, and information security. At the SEI, Alberts leads applied research and development projects in the field of risk management. He has developed practical and innovative methods, tools, and techniques that have been applied by people throughout government and industry organizations, both nationally and internationally. He has also co-authored two books, *Managing Information Security Risks: The OCTAVE^{SM} Approach* (Addison-Wesley 2002) and the *Continuous Risk Management Guidebook* (Software Engineering Institute 1996).

Prior to joining the SEI, Alberts worked at Carnegie Mellon Research Institute, where he developed autonomous robots for hazardous environments, and at AT&T Bell Laboratories, where he helped automate AT&T's manufacturing processes. He has BS and ME degrees in engineering from Carnegie Mellon University.

Julia Allen

Julia Allen is a senior researcher within the CERT® Program at the Software Engineering Institute (SEI), a unit of Carnegie Mellon University in Pittsburgh, PA. Allen's areas of interest include enterprise security governance, operational resilience, and software security and assurance.

Prior to this technical assignment, Allen served as acting director of the SEI for six months and deputy director/chief operating officer for three years. Before joining the SEI, she was a vice president in embedded systems software development for Science Applications International Corporation and managed large software development programs for TRW (now Northrop Grumman).

Ms. Allen is the author of *The CERT Guide to System and Network Security Practices* (Addison-Wesley 2001), the CERT Podcast Series: Security for Business Leaders, and the CERT governance portal. She is a co-author of *Software Security Engineering: A Guide for Project Managers* (Addison-Wesley 2008) and a contributing author to CERT's Resiliency Management Model. Her degrees include a BS in computer science (University of Michigan), an MS in electrical engineering (University of Southern California), and an executive business certificate (University of California – Los Angeles).

Dawn M. Cappelli

Dawn Cappelli is Technical Manager of the Threat and Incident Management Team, which includes the Insider Threat Center, in the CERT Program at Carnegie Mellon University's Software Engineering Institute. Her team's mission is to assist organizations in improving their security posture and incident response capability by researching technical threat areas; developing information security assessment methods and techniques; and providing information, solutions, and training for preventing, detecting, and responding to illicit activity. Her team members are domain experts in insider threat and incident response, and team capabilities include threat analysis and modeling; development of security metrics and assessment methodologies; and creation and delivery of training, courses, and workshops. Cappelli has 29 years of experience in software engineering, including programming, technical project management, information security, and research. Cappelli regularly presents at national and international conferences, and is also adjunct professor in Carnegie Mellon's Heinz School of Public Policy and Management.

Before joining CERT in 2001, Cappelli was the Director of Engineering for the Information Technology Development Center of the Carnegie Mellon Research Institute (CMRI). While with CMRI, she was a technical and program manager for a variety of information networking projects. These projects included the design and development of large-scale databases and Internet-based systems that adhered to data privacy and security requirements, the design and implementation of multi-organizational portals for preparation and response to weapons of mass destruction and collaboration among public health department epidemiologists, and the design and development of a networked media solution for remote collaboration between teachers of children with special needs and consultants.

Previously she worked in Computing Services at Carnegie Mellon University where she led several teams in the areas of web application development, database development, and networked media. Cappelli began her career at Carnegie Mellon University (CMU) in the Information Technology Department at the SEI, managing development of various web and database applications, as well as managing the redesign of the SEI's website by an interdisciplinary project team. Prior to her career at CMU, Cappelli worked for Westinghouse Electric Corporation. While at Westinghouse, she designed and developed systems for nuclear power plants, including real-time graphical user interface systems for power plant operators and computer-aided engineering systems for nuclear plant designers.

Cory F. Cohen

Cory F. Cohen is a senior member of the CERT technical staff, guiding the research and development work of the Malicious Code Analysis team. During his 12 years at CERT, he has worked as an security incident handler, a vulnerability analyst, and a malicious code analyst. His recent work has focused on large-scale automated analysis of malicious code samples collected by CERT.

Prior to joining CERT, Cohen worked for the University of Louisville as HP/UX System Administrator in the engineering school where he managed the primary computing cluster. He also worked for the university as an IDMS/R database administrator maintaining production payroll and student record systems. Cohen holds a BS in information science and data processing from the University of Louisville.

Timothy Daly

Timothy Daly is a Senior Member of the Technical Staff in the Software Engineering Institute at Carnegie Mellon University and is currently serving as a member of the FX/MC (Function Extraction for Malicious Code) team. Prior to this, he served as Research Scientist at the Center for Algorithms and Interactive Scientific Software at City College in New York, where he was the lead developer on Axiom (a General Purpose Computer Algebra System) and Magnus (an Infinite Group Theory system). Formerly he was employed at IBM Research in Yorktown, New York where he participated in projects on rewritable paper, natural language understanding, expert systems, knowledge representation, computer algebra, and industrial robotics. He holds one patent in the area of robotics. He helped develop 4 commercial programming languages. He has published a tutorial book on Axiom and is the lead developer on that open source project. He has taught at City College of New York, Vassar College, and William Patterson College. He holds a Master's degree in Computer Science from Fairleigh Dickinson University and a Bachelor of Science in Mathematics from Montclair State University.

Michael Duggan

Michael Duggan is a software developer for CERT's Network Situational Awareness team. He has been a staff member of CERT since 2003. He has primarily worked on network flow collection and analysis infrastructure.

Prior to joining the SEI, Duggan was a programmer and designer for the Language Technologies Institute at Carnegie Mellon University. Duggan has a BS in Electrical and Computer Engineering from Carnegie Mellon.

Robert J. Ellison

Robert J. Ellison is a senior member of the technical staff in the Networked Systems Survivability Program (NSS) at the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) and a founding member of the SEI.

While at the SEI he has served in both technical and management roles. He participates in the evaluation of software architectures and contributes from the perspective of security and reliability measures. As a member of the NSS program, he is one of the developers of the Survivability Analysis Framework (SAF), has been the project leader for the Build-Security-In project, and currently leads the Supply-Chain Risk project.

Dr. Ellison received his MS and PhD in mathematics from Purdue University. He is a member of the IEEE Computer Society and the Association of Computing Machinery (ACM).

Phillip Groce

Phil Groce is a member of the technical staff in the CERT Program. As a member of the Network Situational Awareness group, he develops tools to help analyze and monitor large networks for security and capacity problems. Before joining CERT, Groce worked at CipherTrust (now a division of McAfee), developing email security solutions. Groce was also a senior software engineer at SecureWorks, an Atlanta-based managed security provider. Groce currently works on improving data visualization capabilities in operational network analysis.

Jeffrey S. Havrilla

Jeffrey S. Havrilla has been a senior member of the technical staff at the Software Engineering Institute for over 10 years, primarily focused on software security engineering. His current area of work is analyzing malicious code and artifacts associated with computer security intrusions.

Havrilla was previously the technical leader of the CERT/CC vulnerability discovery team, part of the CERT/CC focused on finding software vulnerabilities in deployed software using both static and dynamic analytical tools.

Prior to working at the SEI, Havrilla worked at the University of Pittsburgh Medical Center and School of Medicine as a large-scale database, network and research systems administrator and programmer. Havrilla has a Master's of Science in Telecommunications from the University of Pittsburgh School of Information Sciences, and is a member of the IEEE Computer Society and Internet Society (ISOC).

Richard Linger

Richard Linger is manager of the CERT Survivable Systems Engineering group. He directs research and development for the Function Extraction project for software behavior computation, with application to software development and testing and to malware detection and analysis. Linger also serves as a member of the faculty at the Carnegie Mellon University Heinz School of Public Policy and Management. At IBM, he partnered with Dr. Harlan Mills, IBM Fellow, to create Cleanroom Software Engineering technology for development of ultra-reliable software systems, including box-structure specification, function-theoretic design and correctness verification, and statistical usage-based testing for certification of software fitness for use. He has extensive experience in project management; system specification, architecture, design, verification, and certification; software re-engineering and reverse engineering; and technology transfer and education. He has published three software engineering textbooks, 12 book chapters, and over 60 papers and journal articles. He is a senior member of the IEEE and a member of ACM and AIAA.

Howard F. Lipson

Howard F. Lipson is a Senior Member of the Technical Staff in the CERT Program at the SEI. Lipson has been a computer security researcher at CERT for more than 17 years. He is also an adjunct professor in Carnegie Mellon University's Department of Engineering and Public Policy and an adjunct research faculty member at the Carnegie Mellon Electricity Industry Center. He has played a major role in developing the foundational concepts and methodologies necessary to extend security research into the new realm of survivability, and was a chair of three IEEE Information Survivability Workshops. His research interests include the analysis and design of survivable systems and architectures, software assurance, and critical infrastructure protection (in particular, smart grid security and survivability).

Prior to joining Carnegie Mellon, Lipson was a systems design consultant, helping to manage the complexity and improve the usability of leading-edge software systems. Earlier, he was a computer scientist at AT&T Bell Labs. Lipson holds a PhD in Computer Science from Columbia University.

Nancy R. Mead

Nancy R. Mead is a senior member of the technical staff in the CERT Survivable Systems Engineering group. Mead is also a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. She is currently involved in the study of security requirements engineering and the development of professional infrastructure for software engineers.

Prior to joining the SEI, Mead was a senior technical staff member at IBM Federal Systems, where she developed and managed large real-time systems. She also worked in IBM's software engineering technology area and managed IBM Federal Systems' software engineering education department. She has developed and taught numerous courses on software engineering topics.

Mead has more than 150 publications and invited presentations, and has a biographical citation in Who's Who in America. She is a Fellow of IEEE and the IEEE Computer Society and a Distinguished Member of the ACM. Mead received her PhD in mathematics from the Polytechnic Institute of New York, and a BA and an MS in mathematics from New York University.

Samuel A. Merrell

Samuel Merrell is a Member of the Technical Staff on the Resilient Enterprise Management Team at CERT. Merrell works with organizations to improve their information security risk management capabilities. This work includes Critical Infrastructure Protection projects within the Department of Homeland Security and analysis of federal (DoD and civilian agency) information security programs, including Federal Information Security Management Act (FISMA) compliance efforts. Recent projects include assisting in the development of the CERT Resilient Enterprise Framework and evaluating Service Oriented Architecture initiatives within the U.S. Military.

Prior to joining the SEI, Merrell spent seven years as the Information Technology Manager for a Pittsburgh-area community bank. Before that, he was an information technology consultant, primarily support the IBM AS/400. Merrell holds an undergraduate degree from the University of Pittsburgh, the Certified Information Systems Security Professional (CISSP) certification, and a number of SANS certificates, and is currently working towards a master's degree in Information Security at Carnegie Mellon University.

Philip Miller

Dr. Philip Miller is a Senior Member of the Operational Staff at the Software Engineering Institute. Dr. Miller joined the SEI in 2005. He was responsible for launching the Mexican TSP Initiative and the MSIT-SEM program. Recently he has refocused on brining in research funding into the SEI. Prior to 2005, Miller co-founded iCarnegie, Inc., a Carnegie Mellon University educational spin-off. He was a member of the Carnegie Mellon School of Computer Science faculty from 1979 through 1998.

Soumyo D. Moitra

Soumyo Moitra is a Senior Member of the Technical Staff with CERT Network Situational Awareness Group. He has been involved with modeling and analyzing network traffic for security and monitoring. He is currently working on metrics for the cost-effectiveness of network sensors and modeling network security operations.

Prior to his joining the SEI, Soumyo taught Operations Management, worked on telecommunications services and planning at Bellcore, New Jersey, and also taught Policy Analysis at Baruch College, New York. He has an MA from Cornell University, an MS from Syracuse University and a PhD from SUPA (now Heiz College), CMU. He has been an Alexander von Humboldt Fellow at the Max-Planck-Institute, Freiburg, Germany and a Visiting Professor at NTT, Tokyo, Japan. He has published journal articles in a number of subject areas and presents regularly at conferences. He is a member of INFORMS (Institute for Operations Research and Management Science), the American Statistical Association, and SIGMA XI: The Research Society.

Andrew P. Moore

Andrew Moore is a Senior Member of the Technical Staff of the CERT Program at the Software Engineering Institute at the Carnegie Mellon University. Moore explores ways to improve the security, survivability, and resiliency of enterprise systems through insider threat and defense modeling, incident processing and analysis, and architecture engineering and analysis. Before joining the SEI in 2000, he worked for the Naval Research Laboratory investigating high assurance system development methods for the Navy. He has over twenty years experience developing and applying mission-critical system analysis methods and tools, leading to the transfer of critical technology to both industry and the military. Moore received his BA in Mathematics from the College of Wooster and MA in Computer Science from Duke University.

While at the NRL, Moore served as member of the U.S. Defense Science and Technology review (Information Technology TARA) panel on Information Assurance; the International Technical Cooperation Program, Joint Systems and Analysis Group on Safety-Critical Systems (TTCP JSA-AG-4); and the Assurance Working Group of DARPA's Information Assurance Program. He has served as Principal Investigator on numerous projects sponsored by NSA and DARPA. He has also served on numerous computer assurance and security conference program committees and working groups. Moore has published a book chapter and a wide variety of technical journal and conference papers. His research interests include computer and network attack modeling and analysis, IT management control analysis, survivable systems engineering, formal assurance techniques, and security risk analysis.

Rich Pethia

Richard Pethia is the Director of the CERT Program. The program conducts research and development activities to produce technology and systems management practices that help organizations recognize, resist, and recover from attacks on networked systems. The program's CERT Coordination Center (CERT/CC) has formed a partnership with the Department of Homeland Security to provide a national cyber security system, US-CERT. In 2003, Pethia was awarded the position of SEI Fellow for his vision and leadership in establishing the CERT/CC, for his development of the research and development program, and for his ongoing work and leadership in the areas of information assurance and computer and network security. Pethia is also a co-director of Carnegie Mellon University's CyLab. CyLab is a public/private partnership to develop new technologies for measurable, available, secure, trustworthy, and sustainable computing and communications systems. This university-wide, multidisciplinary initiative involves more than 200 faculty, students, and staff at Carnegie Mellon.

Mark Pleszkoch

Mark Pleszkoch is a Senior Member of the Technical Staff at CERT. He is an expert in function-theoretic mathematical foundations of software, and focuses on automation of formal methods. As a member of the function extraction research and development team, he is responsible for creating theoretical foundations and engineering automation for FX systems.

Prior to joining CERT, Pleszkoch worked at IBM for 21 years in various capacities. As a member of IBM's Cleanroom Software Technology Center, he provided education and consultation to clients in software process, software engineering technologies, and software testing. He was the principal architect of the IBM Cleanroom Certification Assistant tool set for statistical testing automation.

Pleszkoch received his PhD in Computer Science from the University of Maryland and an MA and a BA in Mathematics from the University of Virginia. He is a member of the IEEE and the Association for Symbolic Logic.

Stacy Prowell

Stacy Prowell is a Senior Member of the Technical Staff at CERT. He is an expert in the function-theoretic foundations of software, and is currently conducting research and development for function extraction technology. Prowell has managed both commercial and academic software development projects and consulted on design, development, and testing of applications ranging from consumer electronics to medical scanners, from small embedded real-time systems to very large distributed applications.

Prior to joining the SEI in 2005, Prowell was a research professor at the University of Tennessee. To support wider adoption of rigorous methods in industry, he started the Experimentation, Simulation, and Prototyping (ESP) project at the University of Tennessee, which develops software libraries and tools to support application of model-based testing and sequence-based specification. Software developed by this program is in use by over 30 organizations.

Prior to working at the university, he served as a consultant in the software industry. His research interests include rigorous software specification methods, automated statistical testing, and function-theoretic analysis of program behavior. Prowell holds a PhD in Computer Science from the University of Tennessee and is a member of the ACM, IEEE, and Sigma Xi.

Kirk Sayre

Kirk Sayre is an expert in the function-theoretic mathematical foundations that are the basis for function extraction technology. He is currently working on development of the core rewriting engine for the FX system, as well as on formal testing for the system. In addition, Sayre is involved in research involving the application of programming patterns to the development of secure software.

Prior to joining CERT, Sayre was a research professor at the University of Tennessee, where he developed an automated testing framework for the certification of generic scientific computing libraries. In his position at UT, Sayre also developed a CASE tool to support the editing and creation of rigorous sequence-based software specifications. This tool is currently being used on software projects at Oak Ridge National Laboratory and Bosch. Sayre has developed software in many different areas, including educational web applications, automated testing tools, CASE tools, medical devices, and weapons systems.

Robert C. Seacord

Robert C. Seacord leads the Secure Coding Initiative at CERT, located in Carnegie Mellon's Software Engineering Institute (SEI) in Pittsburgh, PA. CERT, among other security related activities, regularly analyzes software vulnerability reports and assesses the risk to the Internet and other critical infrastructure. Robert is an adjunct professor in the Carnegie Mellon University School of Computer Science and at the Information Networking Institute. He represents CMU at PL22.11 (ANSI "C") and is a technical expert for the JTC1/SC22/WG14 international standardization working group for the C programming language.

James F. Stevens

James F. Stevens is a senior member of the technical staff in the Networked Systems Survivability (NSS) Program at Carnegie Mellon University's Software Engineering Institute (SEI). The CERT Coordination Center is also part of this program. As a part of the Survivable Enterprise Management group, Stevens is responsible for developing methods, tools, and techniques for resilient enterprise management. This work includes designing and delivering various information security risk assessment, analysis, and management technologies for customers in the government and the private sector. Stevens has been working in information security for over fifteen years and holds a BS degree in Electrical Engineering from the University of Notre Dame and an MBA from Carnegie Mellon University's Tepper School of Business. Stevens holds the CISSP certification as well.

Ed Stoner

Ed Stoner is a Member of the Technical Staff in the Network Situational Awareness Group at CERT. He holds a BS in Math/Computer Science from Carnegie Mellon. Prior to joining the NetSA team, Ed was the network engineer for the CERT Infrastructure Team. His research interests include network protocols and large-scale DNS trends.

David Svoboda

David Svoboda is a software security engineer at CERT, at the Software Engineering Institute (SEI) in Pittsburgh, PA. David has been the primary developer on a diverse set of software development projects at Carnegie Mellon since 1991, ranging from hierarchical chip modelling and social organization simulation to Automated Machine Translation (AMT). His KANTOO AMT software, developed in 1996, is still (as of 2008) in production use at Caterpillar. David is also actively involved in several ISO standards groups: the JTC1/SC22/WG14 group for the C programming language, and the JTC1/SC22/WG21 group for C++.

Mark Thomas

Mark Thomas is a Member of the Technical Staff with the Network Situational Awareness Group, where he works as a software engineer, primarily developing tools for network flow collection and analysis and responding to user feedback.

Although Thomas's degrees are in Chemical Engineering (a BS from West Virginia Tech, an MS and a PhD from Carnegie Mellon), he has always enjoyed programming, and his PhD work on computer environments for engineering design allowed him to join his programming skills with his formal education. Thomas worked on a variety of software projects either at Carnegie Mellon or near its campus in the years between his graduation and his joining CERT in 2003.

Randy Trzeciak

Randy Trzeciak is currently a Member of the Technical Staff for the Software Engineering Institute's (SEI) Networked Systems Survivability (NSS) Program.

Mr. Trzeciak is a member of a team in CERT focusing on insider threat research, including insider threat studies being conducted with the U.S. Secret Service National Threat Assessment Center, DoD's Personnel Security Research Center (PERSEREC), and Carnegie Mellon's CyLab. The studies analyze the physical and online behavior of malicious insiders prior to and during network compromises. Other insider threat research uses system dynamics modeling for risk analysis of the impacts of policy decisions, technical security measures, psychological issues, and organizational culture on insider threat. Mr. Trzeciak also is an adjunct professor in Carnegie Mellon's H. John Heinz School of Public Policy and Management.

Prior to his current role in the NSS Program, Mr. Trzeciak managed the Management Information Systems (MIS) team in the Information Technology Department at the SEI. Under his direction, the MIS team developed and supported numerous mission critical, large-scale, relational database management systems. The applications developed supported business functions across all levels in the SEI, from the Director's Office to administrative support staff.

Prior to his time working at the SEI, Mr. Trzeciak was a software engineer for the Information Technology Development Center of the Carnegie Mellon Research Institute (CMRI), responsible for a variety of information networking projects. These projects included the design and development of large-scale databases and internet-based systems that adhered to data privacy and security requirements; the design and implementation of multi-organizational portals for preparation and response to weapons of mass destruction; and collaboration among public health department epidemiologists.

Previously he worked in Computing Services at Carnegie Mellon University where he was a lead developer and database administrator on several teams focusing on the areas of web application development, database development, and networked media.

Prior to his career at Carnegie Mellon, Mr. Trzeciak worked for Software Technology, Incorporated (STI) in Alexandria Virginia. For 9 years, Mr. Trzeciak was a consultant to the Naval Research Laboratory (NRL) working on numerous projects designing, building, and supporting large-scale relational database management systems. During his employment with STI, Mr. Trzeciak also filled the role of Information Systems Business Manager. Mr. Trzeciak was responsible for managing business operations for the Information Systems Segment within STI.

Mr. Trzeciak holds a MS in Management from the University of Maryland, a BS in Management Information Systems, and a BA in Business Administration from Geneva College.

Rhiannon Weaver

Rhiannon Weaver is a member of the Technical Staff for the Network Situational Awareness Group. She holds a BS in Mathematics and a BS in Computer Science from Penn State University, and a MS in Statistics from Carnegie Mellon University, where she is also pursuing her PhD in statistics.

Weaver provides support for advanced modeling techniques for network anomaly detection and large-scale trending of Internet-wide phenomena. Her research interests include time series analysis of network data, data collection and inference in hierarchical and Bayesian models, as well as addressing the challenges of evaluating and applying advanced modeling and data mining techniques in operational environments.

David W. White

David White is a Senior Member of the Technical Staff at CERT. White is responsible for developing and implementing strategies that lead to the widespread dissemination and use of methods, techniques, and tools that help organizations manage information security risks. He is also a member of the development team for the CERT Resiliency Engineering Framework, a process improvement framework that provides guidelines for improvement framework that provides guidelines for managing security and business continuity from an enterprise risk management perspective.

White has a bachelor's degree in Civil Engineering and Public Policy from Carnegie Mellon University and a master's degree in Civil Engineering with a specialization in robotics from Carnegie Mellon University. He is currently based in New York City.

Evan Wright

Evan is an analyst for the Network Situational Awareness Team (NetSA). Evan's research interests include next-generation technologies, network design, routing protocols, and design of network attack tools.

Prior to joining the SEI, Wright completed graduate school at Carnegie Mellon, where he obtained his MS in Information Security and Technology Management from the School of Engineering. He also holds a BS in Technology Systems from East Carolina University.

Wright worked as a Network Administrator at ABC Phones in North Carolina and as a consultant for various other companies. Evan holds the Cisco Certified Networking Professional certificate and four other IT certifications.

Bradford J. Willke, CISSP

Bradford Willke is a member of the Survivable Enterprise Management group and a senior member of the CERT technical staff. Willke is responsible for leading the Information and Infrastructure Protection team, and conducts research, development, and process improvement activities in risk, threat, and vulnerability management methodology related to information security management. Willke also leads projects to develop strategies and provide support for national and international critical infrastructure protection initiatives. In addition, he worked on the development of the SEI's principle risk assessment methodology, the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE^{SM}) Method.

Before joining the SEI, Willke was a technical intern with Southern Illinois University at Carbondale, where he installed, managed, and maintained the university's first firewall, which protected the university's multi-million dollar Oracle investment. He also provided technology and security management for computing resources of the 90th Security Police Squadron, Francis E. Warren Air Force Base, Wyoming. Willke served in the United States Air Force as a law enforcement specialist and organizational computer security officer from 1993-1997.

Willke holds a professional certificate in information protection and security from the University of New Haven, and received a BS in information systems technologies from Southern Illinois University at Carbondale in 1999. He received an AAS in criminal justice from the Community College of the Air Force in 1997, and has been a Certified Information System Security Professional (CISSP) since 2004.

Carol Woody

Carol Woody is a senior member of the technical staff at SEI. She leads a team of researchers focused on building capabilities in defining, acquiring, developing, measuring, managing, and sustaining networked systems and software so as to exhibit desired security attributes over time.

Woody has over 25 years of experience in software development and project management covering all aspects of software and systems planning, acquisition, design, development, and implementation in large complex organizations. Woody has a biographical citation in Who's Who in American Women and Who's Who in Finance and Industry. She is a senior member of IEEE and ACM, and a member of PMI.

Dr. Woody holds a BS in mathematics from The College of William and Mary, an MBA with distinction from Wake Forest University, and a PhD in Information Systems from NOVA Southeastern University.

Justin Zhan

Dr. Justin Zhan is a faculty member at Carnegie Mellon University and a research director of the privacy, security, and decision informatics lab at Carnegie Mellon CyLab Japan. His research interests include the privacy and security aspects of data mining, privacy and security issues in social networks, privacy-preserving scientific computing, privacy-preserving electronic business, artificial intelligence applied in the information security domain, data mining approaches for privacy management, and security technologies associated with compliance and security intelligence. He has served as an editor/advisory/editorial board member for many international journals and a committee chair/member for over 80 international conferences. He has published over one hundred articles in various peer reviewed journals and conferences.

CERT

Software Engineering Institute
Carnegie Mellon

The 2009 CERT Research Annual Report was produced by SEI Communications.

Executive Editor
Richard Linger

Design
Robert Fantazier

Production
David Biber
David Gregg
Melissa Neely